パソコンプログラマのためのクリエイティブマガジン

1991年4月1日発行(毎月1回18日)　第1巻　第4号　通巻4号

月刊プログラマーズ・ページ

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1991 4 APRIL

定価980円

特集 C読本

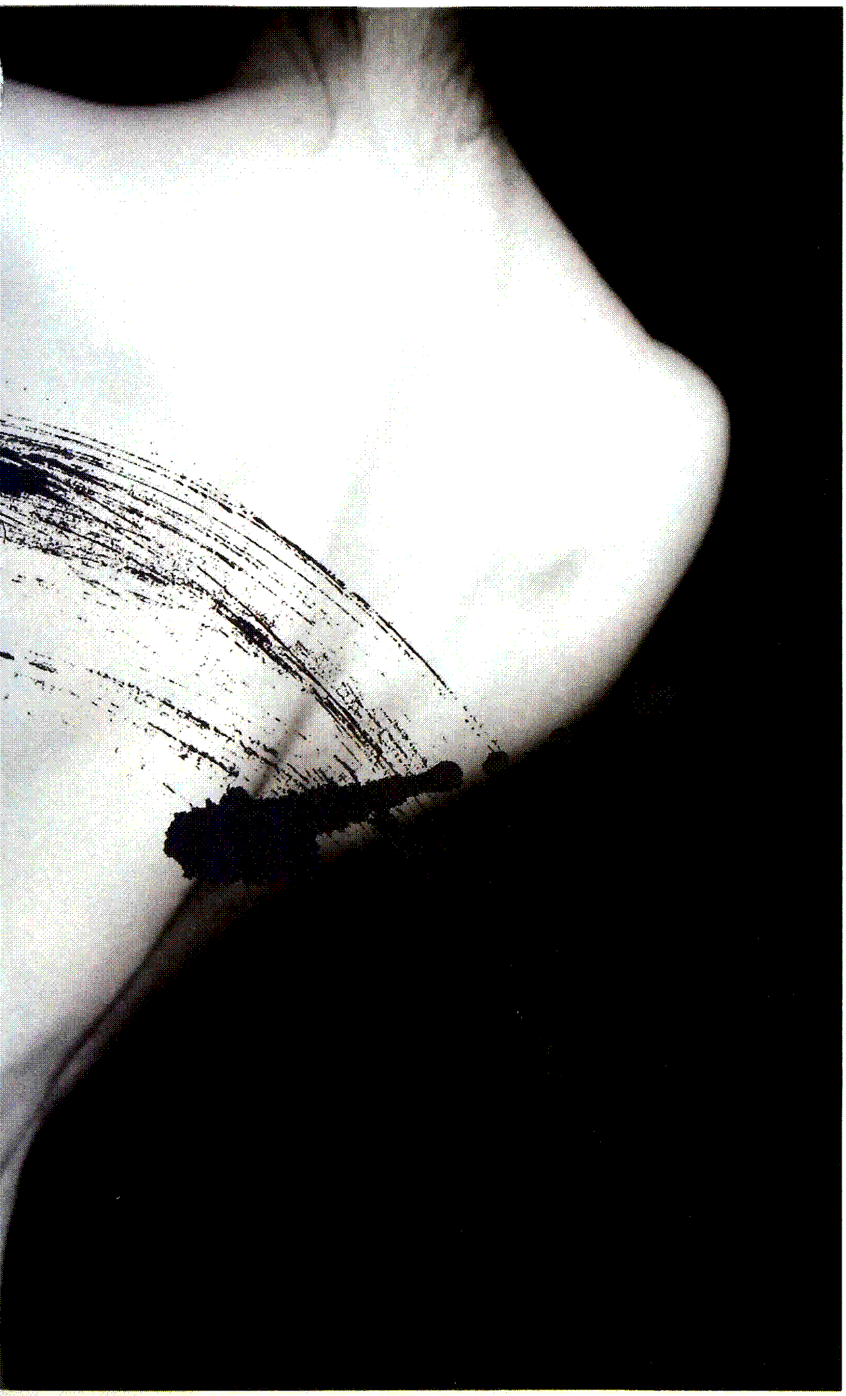
C言語と正しく付き合うためのポイント

付録:5"2HDフロッピーディスク

■本誌掲載プログラム／綴じ込み付録テキストファイル(Turboシリーズプログラム起動オプション)／クイズ・ランダムスクリーン パズルゲーム(PC-9801)／高圧縮書庫管理プログラムLHa

SE
Shoeisha Co., Ltd.

美しさを追求する
パーソナルコンピュータはAXです。



世界の標準機を、日本標準機に。我々が、パソコンの新しい時代を拓きます。

Acer
Canon
CASIO
HITACHI
KYOCERA
MITSUBISHI
NCR
OKI
olivetti
SANYO
SHARP
SONY
UNISYS
YHP

アルファベット順

画面表示の美しさなら、AXパソコンです。高解像度のグラフィックと質の高いフォント表示、そして海外の最新マルチメディア環境の導入。AXパソコンを使うことにより、美しく斬新なプレゼンテーションが可能になります。パソコンに美しさを求めるあなたに、最もふさわしい1台です。AXパソコンは、事実上の世界標準機であるPC/ATとその互換機の世界をベースにしたパソコンの共通仕様。海外からの最新データを迅速に利用することにより、つねに新鮮なプレゼンテーションができるのです。
メーカーが違えば互換性のなかった、これまでのパソコンの世界。そこに、「パソコンはもっと使いやすくなくては」というユーザー本位の発想を持ちこんだのが、AXです。メーカー間の、そして国際間の壁を取り払い、パソコンの新しい世界を築くために。AX協議会会員558社(90年12月現在)は、それぞれの技術を結集しているのです。

THE ARCHITECTURE EXTENDED—AX

〈資料請求番号P02〉

386|DOS-Extender!

386|DOS-Extenderは、80386CPUをプロテクトモードに切り替えることによって従来のMS-DOS環境のままで、ユーザプログラムがMS-DOSの640Kバイトのメモリ制限を破り、実装されているメモリを全て連続した記憶域として利用できるようにします。さらに386|DOS-Extenderは80386CPU本来の実行速度や4Gバイトのアドレス空間をユーザプログラムに提供するので、これを利用すると従来のMS-DOSプログラムより巨大なメモリを使用する実行速度の速いプログラムを作成・実行することができます。386|DOS-Extenderは386|TOOLBOXに含まれているユーティリティです。

Phar Lap社386|TOOLBOX

386|TOOLBOXは米国Phar Lap社の開発した386|ASM、386|LINK、386|LIB、RUN386(386|DOS-Extender)、MINIBUG、CFIG386で構成されています。これらはアセンブラ、リンカ、ライブラリアン、386|DOS-Extender、デバッガ、及びそれらの環境カスタマイズユーティリティであり、80386CPUのプロテクトモードプログラムの開発環境を提供します。

386|ASM

386|ASMはIntel社の80386CPUのためのアセンブラです。386|ASMは、8086、186、286、486CPUや8087、287、387内蔵FPUのための命令もアセンブルすることができます。8086、186、286CPUのための命令をアセンブルするとIntel社のオブジェクトモジュールフォーマット(OMF-86)を生成します。386、486CPUの命令に対しては、Phar Lap社独自のオブジェクトファイルフォーマットを生成します。386|ASMにはリアルモードで動作する386|asm.exeと、プロテクトモードで動作する386|asmp.exeの2つが用意されています。386|asmは、動作するCPUを問わず、どのMS-DOS上でも使用することができます。386|asmpは386、486CPU上でしか動作しませんが、プロテクトモードで動作するため、アセンブル動作も高速であり、内部のシンボルテーブルなどにサイズ制限がありませんので、大きなプログラムのアセンブルに適しています。またコマンドラインスイッチにより286以降のCPUに対しては、リアルモード用とプロテクトモード用の両方に対応するプログラムをアセンブルすることができます。

386|LINK

386|LINKはIntel社の386、486CPU用のリンカです。386|LINKは、アセンブラやコンパイラで生成された別々のオブジェクトモジュールやライブラリをリンクし、ロードモジュールを生成します。386|LINKは、オプションスイッチにより386CPUのプロテクトモードのプログラム(.EXPファイル)を作成したり、8086CPUのリアルモードのプログラム(.EXEファイル)を生成することもできます。また386|LINKはPhar Lap社の386|DEBUG(別売デバッガ)で使用できる外部シンボルに関する情報をロードモジュールの中に付加する機能を持っているため、386|DEBUGによるデバッグ作業を容易にすることができます。386|TOOLBOXには386|LINKの他に、FASTLINKというリンカが含まれています。FASTLINKは80386、486CPUのプロテクトモードで動作するリンカです。FASTLINK、386|LINKとも同じ動作をしますが、386|LINKはMS-DOSの640Kバイトのメモリ制限のた

32bit専用プログラム開発支援シリーズ

HIGH C386

MetaWare社のHIGH C386は80386CPUプロテクトモード用最適化コンパイラです。

HIGH C386は、80386CPUのプロテクトモード上で動作するハイレベルのCプログラムを構築するプロフェッショナルソフトウェア技術者のために設計されました。正確なエラーメッセージと優れた特長を兼ね備え、また、ソフトウェア開発者は、多種多様なトグルスイッチとプラグマを活用することによって、開発者のニーズに合わせてコンパイラの初期設定や使用の変更ができます。

HIGH C386は、386|TOOLBOXに含まれる386|DOS-Extenderとの組み合わせにより、80386CPUプロテクトモードの膨大なアドレス空間と高速な演算処理速度を活用できるようになります。

特長

- コンパイラは、ANSI Cの仕様を満たしており、必要ならばコンパイルオプションやpragma、トグルスイッチ等を用いてコンパイラ自体の使用を変更して、他のCコンパイラと互換性を持たせられます。
- 各種数値演算コプロセッサ(80287、80387、Weitek1167 3167)をサポートしており、高速な演算処理を実現しています(ベンチマークテスト表を参照)。
- LINTのようなチェックにより正確さと設定診断を得られます。
- ソースでの注釈付きのアセンブリリストもしくはファイルを生成できます。
- Ada言語等の名前付けによるパラメータ指定が可能です。
- さまざまな最適化処理を駆使し、より高速のコード生成を図っています。
  最適化処理の一例：
  デッドコードの除去/定数の複写伝播/定数式の畳み込み
  ローカルとグローバルの共通部分式の削除/ループ内の不変値式の移動
  テールリカージョン(末端再帰)の除去/グローバルなレジスタの割当
  関数のインライン展開

ベンチマークテスト

WHETSTONE(自社計測)		
使用機器：PC-9801RL20MHz＋数値演算コプロセッサ(80387 or Turbo-3167)		
emulator＋noFPU	58	[K whetstone/sec]
emulator＋80387	200	[K whetstone/sec]
80387 inline code	1540	[K whetstone/sec]
Turbo-3167 inline code	2380	[K whetstone/sec]
DHRYSTONE(ソフトバンク(株)刊『月刊Cマガジン』'90.8月号より抜粋)		
使用機器：PC-9801RA2 数値演算コプロセッサ(80387)		
HIGH C386 6250 [dhrystone/sec] 8(SEC) for 50000 PASSES		

動作環境

●必要なハードウェア

〈本体〉

NEC PC-9800シリーズ(XL2、RA、RL、LS、ES、RA21/51、RS21/51、DA、DS、NS、T)

NEC PC-98シリーズ(Model 70、100)

EPSON PC-386シリーズ(386、LS、V、M、G)

東芝 J-3100(SGT、SGX、GXS)、J-3300

IBM PC/AT互換機、各種AXマシン(80386、80386SX搭載機)

〈ハードディスク〉

1MByte以上のプロテクトモードメモリ

●必要なソフトウェア

〈MS-DOS〉

NEC、EPSON機：MS-DOS Ver3.3以上

IBM PC/AT機、AX機：PC-DOS Ver3.2以上

東芝 J-3100等：日本語DOS Ver3.1以上

英語DOS Ver3.3以上

〈Phar Lap社 386|TOOLBOX〉(弊社取扱製品)

NDP FORTRAN386

NDP FORTRAN386は、FORTRAN77仕様に完全準拠した80386CPUプロテクトモード専用FORTRANコンパイラです。

最大の特長である、プロテクトモードの4GByteのアドレス空間と、最適化による高速コードの生成により、VAX8600/SUN3等の汎用コンピュータ/ワークステーションクラスの規模と速度を持つ、アプリケーションの開発・実行を可能にしました。

特長

- プロテクトモードのアドレス空間4GByteを活用可能。1MByteを超えるプログラムや配列の実行・定義ができ、上限はパソコン本体に実装されたプロテクトモード用メモリの量にのみ制限されます。
- 数値演算コプロセッサの最適コードの生成、インラインコードの生成といった最適化処理を搭載し、プログラムのより高速な実行を可能にします。
- 数値演算コプロセッサ80387をサポート。さらに、80387の約3倍(浮動小数点テスト時)の性能を持つ、Weitek1167/3167チップ、Turbo-3167ボードをサポート。NDP FORTRAN386で作成された計算ルーチンの速度をより向上させられます。
- FORTRAN77仕様のフルセットコンパイラです。また、拡張機能として、ANSI X3.9-1987とDODのMIL STD 1753さらに、UNIX4.2BSDにおけるf77の拡張機能を完全に満たしています。
- R/M FORTRANとVAX FORTRANからの移植を容易にするための拡張機能をサポートしています。
- シンボル名として31文字まで使用できます。$の使用も可能です。
- INCLUDEのネストが可能です。
- MIL STD-1753のビット操作をすべてサポートしています。IOR、IAND、IEOR、NOT、ISHFT、ISHFTC、ISHFTL、ISHFTR、BTEST、IBSET、IBCLR、MVBITS。
- ベッセル関数、ガンマ関数、誤差関数をサポートしています。
- ZとO編集が入出力に使用できます。
- FORTRAN66のDOループをオプションスイッチにより選択できます。
- カンマを使用したフリーフォーマット入力をサポートしています。
- $を入出力の改行文字として利用できます。
- カラム1にX、Dを記入することにより、条件付きコンパイルが可能です。
- COMMONとEQUIVALENCEにおいて文字と数字を混在できます。

めに、リンク中に内部のシンボルテーブル領域が不足する可能性があります。しかしFASTLINKはプロテクトモードで動作するために640Kバイトの制限がなくなるので、より大きなプログラムを高速にリンク処理することができます。

386|LIB

386|LIBはIntel社のOMF-86とPhar Lap社独自のOMFフォーマットのオブジェクトファイル用のライブラリアンです。386|LIBは386|LINK等で使用できるライブラリファイルを作成することができます。

MINIBUG

MINIBUGはMS-DOSの標準デバッガ・DEBUGに準じており、リアルモードとプロテクトモードのプログラムのどちらもデバッグすることができます。

(MINIBUGの基本的な機能)

- 80386、486CPUの各レジスタの内容を参照、修正することができる。
- メモリの内容を参照、修正することができる。
- 同時に最大4ヶ所までのブレークポイントを設定することができる。
- プロテクトモード管理用のレジスタやテーブルの表示ができる。

CFIG386

CFIG386は386|TOOLBOX内の各ツールのオプションスイッチパラメータをカスタマイズするためのプログラムです。CFIG386を使用することにより、各ツールの起動時にいちいち長いオプションスイッチを指定しなくても、オプションを指定した状態で起動させることができます。CFIG386でカスタマイズできるツールは上記までに紹介したRUN386、386|ASM、386|ASMP、386|LINK、FASTLINK、386|LIB、MINIBUGと別売の386|DEBUGの全てが対象となります。

動作環境

●必要なハードウェア

<本体>

NEC PC-9800シリーズ(XL2, RA, RL, LS, ES, RA21 51, RS21 51, DA, DS, NS, T)
NEC PC-98シリーズ(Model 70, 100)
EPSON PC-386シリーズ(386, LS, V, M, G)
東芝 J-3100(SGT, SGX, GXS), J-3300
IBM PC AT互換機、各種AXマシン(80386, 80386SX搭載機)

<ハードディスク>

1MByte以上のプロテクトモードメモリ

●必要なソフトウェア

<MS-DOS>

NEC, EPSON機：MS-DOS　Ver3.3以上
IBM PC AT機, AX機：PC-DOS　Ver3.2以上
東芝 J-3100等：日本語DOS　Ver3.1以上
英語DOS　Ver3.3以上

- つぎの型が使用できます　REAL*4, REAL*8, INTEGER*1, INTEGER*2, INTEGER*4, LOGICAL*1, LOGICAL*2, LOGICAL*4, COMPLEX*8, COMPLEX*16　デフォルトのLOGICAL, INTEGERは4ですが、それを2に変更可能です。
- コマンドラインに指定したgetarg, iargc, getenvを読み込む機能をサポートしています。

動作環境

●必要なハードウェア

<本体>

NEC PC-9800シリーズ (XL2, RA, RL, LS, ES, RA21/51, RS21/51, DA, DS, NS,T)
NEC PC-98シリーズ (Model70, 100)
EPSON PC-386シリーズ(386, LS, V, M, G)
東芝 J-3100 (SGT, SGX, GXS), J-3300
IBM PC/AT互換機　各種AXマシン(80386, 80386SX搭載機)

<ハードディスク>

2MByte以上のプロテクトモードメモリ
数値演算コプロセッサ(80387またはWeitek3167/TURBO-3167)

●必要なソフトウェア

<MS-DOS>

NEC, EPSON機:MS-DOS　Ver 3.3以上
IBM PC/AT機, AX機:PC-DOS　Ver 3.2以上
東芝　J-3100等:日本語DOS　Ver 3.1以上
英語DOS　Ver 3.3以上

<Phar Lap社 386|TOOLBOX> (弊社取扱製品)

SPACE II

科学技術演算ライブラリSPACE IIはNDP FORTRAN386で使用できる、汎用科学技術演算ライブラリです。

数値演算に必要となるサブルーチンをもれなく集成した汎用科学技術演算ライブラリです。学会論文・雑誌論文などに発表された最新のアルゴリズムを採用。さらに、数値計算上のさまざまな工夫をこらし、計算機の計算精度のダイナミック管理を行うことにより、大幅な計算精度の向上を実現しています。SPACE IIは、科学者・技術者のための熟成されたプログラム開発環境を提供します。

特長

- ワークステーション業界で定評のあるSPACE IIを、MS-DOS上のNDP FORTRAN386用に移植。
- 行列・連立一次方程式・固有値・方程式・積分・常微分方程式・最小二乗法・乱数・補間と微分・多倍長演算・ソート・基礎統計・分布・等高線・高速フーリエ変換など、通常の数値計算に必要な計算分野をすべて網羅しています。
- 精度に関係するサブルーチンは、単精度・倍精度の両方が用意されているので、処理速度重視・精度重視のいずれの使い方もできます。また92桁の浮動小数点演算を可能にする多倍長演算のサブルーチンが特別に用意されています。
- 計算機の計算精度を表すマシンイプシロンの概念を導入しているので、収束演算などで計算精度を特に指定しない場合は、自動的にその計算機の持つ最大の計算精度で計算します。
- SPACE IIはライブラリ形式で提供されるので、使用法はいたって簡単です。個々のサブルーチンの使用法に従ってプログラムを書き、それをコンパイルし、リンク時にSPACE IIをライブラリ指定するだけで実行形式のプログラムができあがります。

386|VMM

386|VMMは386|TOOLBOXに含まれる386|DOS-Extenderに、ハードディスクを利用した仮想記憶機能を付加する仮想記憶マネージャです。これを使用すると、今まで本体に実装されたメモリの大きさに制限されて、実行させられなかったプログラムを実行可能にします。

動作環境

●必要なハードウェア

<本体>

NEC PC-9800シリーズ (XL2, RA, RL, LS, ES, RA21/51, RS21/51, DA, DS, NS,T)
NEC PC-98シリーズ (Model70, 100)
EPSON PC-386シリーズ(386, LS, V, M, G)
東芝 J-3100 (SGT, SGX, GXS), J-3300
IBM PC/AT互換機　各種AXマシン(80386, 80386SX搭載機)

<ハードディスク>

1MByte以上のプロテクトモードメモリ

●必要なソフトウェア

<MS-DOS>

NEC, EPSON機:MS-DOS　Ver 3.3以上
IBM PC/AT機, AX機:PC-DOS　Ver 3.2以上
東芝　J-3100等:日本語DOS　Ver 3.1以上
英語DOS　Ver 3.3以上

<Phar Lap社 386-TOOLBOX> (弊社取扱製品)

価格表

386|TOOLBOX	¥148,000	NDP C386	¥148,000
386|VMM	¥86,000	HIGH C386	¥148,000
386|DEBUG	¥58,000	SPACE II	¥98,000
NDP FORTRAN386	¥148,000		

※すべての価格に消費税は含まれていません。

TSS
テックソフト&サービス株式会社
コンピュータ事業部　〒273 千葉県船橋市本町7-11-5 レランドセンタービル5階
TEL (0474)23-1322　FAX (0474)23-0202

BORLAND

FAST RUNNER

ムダな動きの少ない走りは、ゴールへの道程を確実に短縮する—。
AP完成まであなたをスムーズに導くTURBO PASCAL バージョン6.0、新登場。

■新登場・Turbo Vision
もはや、アプリケーションを書く際にユーザインターフェース等の細かい部分で苦労する必要はありません。新しいTurbo Pascal 6.0に付属する、オブジェクト指向のユーザインターフェース・クラスライブラリTurbo Visionを使えば、プロ開発者レベルのすばらしいユーザインターフェースを持つアプリケーションを、驚くほど簡単に記述することができます。これによって、位置・サイズを自由に変更可能なオーバーラップウィンドゥやプルダウンメニュー、ダイアログボックス、ボタン等を駆使したマウス操作のイベントドリブン・アプリケーションを短時間で作成できます。さらに電卓、カレンダー、エディタ、時計、ディレクトリブラウザ、フォームといったアプリケーションをサンプルとしてソースコード形式で提供。これらを部品としてアプリケーションに組み込むことができます。事実、Turbo Pascal 6.0の新・統合開発環境もTurbo Visionを使って開発されています。つまり、操作性では定評のある統合開発環境と同レベルのアプリケーションが簡単に作成できる訳です。

■大きく改善された統合開発環境
新しい統合環境は、オーバーラップマルチウィンドゥをサポートし、マウス操作も可能です。エディタは、1MBまでのファイルを同時に複数オープンできます。また、内蔵デバッガでは条件ブレークポイント機能やレジスタウィンドウも追加されています。

■組み込みアセンブラ
従来のインラインアセンブル機能に加え組み込みアセンブラが使用できます。ソースプログラムにニーモニックを直接記述できるため、PASCALプログラム中でも簡単にアセンブラ言語を利用できます。

■ハイレゾルーションモードに対応
統合環境のハイレゾ31行モードでの動作、BGIやTurbo Visionでのハイレゾモードをサポート。また、H98シリーズの拡張アトリビュートにも対応しています。

■Turbo Pascal 6.0 Professional
Professional版には、タフなバグを始末するTurbo Debugger 2.01、プログラムのボトルネックを取り除くTurbo Profiler 1.0、MASMコンパチブルな高速アセンブラTurbo Assembler 2.0に加えて、プロテクトメモリを利用して巨大アプリケーションがコンパイルできるコマンドラインコンパイラが付属します。

■Turbo Pascal 6.0の動作環境
機種：PC-9801シリーズ、PC-286/386シリーズ（ハイレゾモード対応、LTは除く）／メモリ：640KB以上／MS-DOS：バージョン2.11以上／マウス：バスマウス

■価格：Turbo Pascal 6.0……………¥29,800
Turbo Pascal 6.0 Professional………¥49,800

TURBO PASCAL® 6.0

※All Borland Products are trademaks or registered trademarks of Borland International, Inc. ©Borland International, Inc.
※その他、商品名は一般に各社の商標です。
●製品に関するお問い合せは、総販売元㈱マイクロソフトウェア アソシエイツ TEL 03-3486-1411までお願いします

株式会社ボーランドジャパン
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400㈹

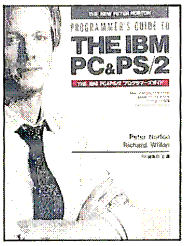
〈資料請求番号002〉

SE 翔泳社が贈る
プログラミングブックス

QuickBASIC スタイルブック

イント・ツーワン著
定価2,300円

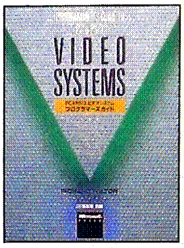
少しでもわかりやすく、質の高いプログラミングを書くために一問一答式の問題集形式になっている「良いプログラミング」のための学習書です。



THE IBM PC & PS/2 プログラマーズガイド

●Peter Norton & Richard Wilton著
●定価5,200円

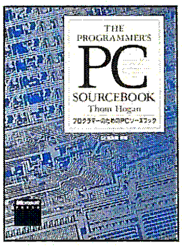
あのピーター・ノートンのベストセラー。IBMプログラマー必携のスタンダードガイダンス。



PC & PS/2 ビデオシステム プログラマーズガイド

●Richard Wilton著
●定価6,000円

IBM PCファミリーのビデオシステム完全ガイド。



プログラマーのためのPCソースブック

●Thom Hogan著
●定価5,200円

IBM PC, PS/2プログラマ必携の技術資料完訳。

Quick C コンパイラVer. 2.0 ランタイムライブラリリファレンス

QuickBASIC Ver. 4.5 ステートメント&関数リファレンス

●マイクロソフト株式会社 監修
●定価各2,400円

マイクロソフトの超人気言語ソフトQuickシリーズ。Quick BASIC Ver. 4.5 Quick C Ver. 2.0スタンダードリファレンス。マイクロソフト社監修でオンラインヘルプを完全書籍化サンプルプログラム満載のリファレンス。

●田中正造 著

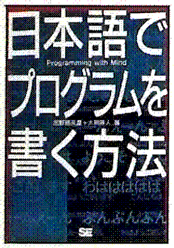
●定価2,000円

BGI／ROM BIOS／EMSの実践テクニックガイダンス、Turbo C 2.0の実践的な機能を豊富な図表と実践的なサンプルプログラムで解説。

QuickBASIC Ver. 4.5 プログラミングブック

●塚本吉彦 著
●定価2,400円

QuickBASICで気軽にプログラミング。BASICビギナーだけではなく昔からのBASICファンにも贈るQuickBASICの入門書です。

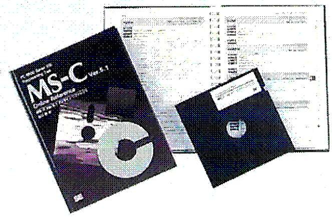


日本語でプログラムを書く方法

●加野島英渡・大網呼人 著
●定価1,200円

本書には、記号、数式は一切出てきません。でもプログラミングの本なのです。日本語で書いたプログラムが本当にパソコンで動きます。

ディスク付MS-Cリファレンス



MS-C Ver. 5.1 オンラインリファレンス

●谷口崇 著
●定価2,200円

フロッピーディスク付のプログラミングツール。

SE 株式会社 翔泳社
東京都千代田区平河町2-4-14 平河町KSビル 〒102
PHONE 03-3263-0447 FAX 03-3263-4526

〈資料請求番号003〉

Brief

Full Screen Editor 日本語版

全米のプログラマが認めた実力！
アメリカでは「ブリーフ」が
プログラミング用エディターとして
最も多く利用されています。

C言語で組めるマクロ！

V.3.0 Version UP! いよいよ登場。
プログラミングエディター新世紀。

よりパワフルで、よりエレガントな プログラミングのために……。

●C言語で組めるマクロ 新機能

使い慣れているC言語でマクロが組めますから、習得が簡単でカスタマイズも思いのままです。

●マクロのスクリーン・デバッガ 新機能

ソースコードでデバックできます。もちろんブレークポイントやウォッチ・ドッグもセットできます。

●テンプレート機能〈構文の自動生成〉 新機能

C、Ada、Basic、COBOL、FORTRAN、MODULA2、Pascalで構文を自動生成します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元にもどすことができますので、やり直しの面倒を気にせず、安心して編集できます。

●安全編集

Briefは、一定時間キーを操作しないこと最新の編集内容をファイルに自動保存します。だから編集中に電源が切れても安心です。

●その他の主な新機能

キー操作のファイルへのセーブとリプレイ、処理可能行数を拡張、リジューム機能、Ctrlキー・アサイン（オプション）、メニュー（オプション）etc。

アメリカでも日本でも高い評価

●プログラミング向きエディターを探している人は、もう探す必要はない。（Jerry Pournelle / BYTE）
●ソフト開発には強力なツール、300回のアンドゥは大きな特徴。（日経バイト）
●既存のエディターに不満を覚えたら、これしかない。（Oh! FM）
●まるで夢のようなアンドゥだ——おもわず間違うのを期待してしまいそうだ。（The C Users Journal）

■主な機能

◆55ウィンドウ◆無限のファイル◆自動コンパイル◆自動エラー・ジャンプ◆自動インデント◆自動テンプレート◆バッファ書出/切換/削除/リスト表示◆ファイル名の指定/表示/変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク/始・終点の入換◆ブロックのカット/コピー/ペースト/印刷/保存/左右へのシフト◆対応する括弧へ移動◆C言語のコメント化/非コメント化◆DOSコマンドの実行◆キー操作記憶/実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価 48,000円
（消費税は含まず）

■5機種対応パッケージ

（東芝、NEC、AX協議会、IBM、富士通）

●5'2DD、3.5'2DDデュアルメディア●MS-DOS3.1以降に対応●英語DOSでは使用できませんが、英語版BRIEFもセット販売いたしております。●お急ぎの方、上記以外のメディア希望の方、英語BRIEFのセット販売希望の方はユーザー・サポート係までご連絡下さい。

日本システム株式会社

●お問合せは 東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL0425-23-4469 FAX0425-27-6127

〈資料請求番号004〉

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

パソコンプログラマのための
クリエイティブマガジン
月刊プログラマーズ・ページ

1991 APRIL No.4

CONTENTS

特集
C読本
C言語と正しく付き合うためのポイント

1 Cを知らないあなたのための
体で覚えるC言語入門 種村明宏 2
2 ワークステーションからパソコンへ
UNIXのC, MS-DOSのC 辻村 浩 15
3 読みやすいプログラムとは?
Cプログラミングのスタイルを決める 秋山 登 25
4 うまくいかないこともある!?
C言語のデバッグテクニック 平賀礼二 37
5 C言語の明日
オブジェクト指向Cプログラミング 石川雅基・神林 靖 47
C言語関連書籍一覧 61

Illustration: NAOKI SATO

表紙イラスト：日暮修一
表紙AD：佐々木幹夫(SE)

C O N T E N T S

パソコン博物館　第4回

Z80成功物語の嚆矢／TRS80　井戸禎光　66

プログラマの道具学　第4回

キーボード　祐安重夫　68

Book Review「オブジェクト指向入門」中村正三郎　118

本のアンテナ　古田島義和

佐々木哲のニューヨークシステム通信　120

西川和久の希望的観測　122

コンピュータの法律　熊谷信太郎　124

アキバ短信

プログラマの仕事場 第1回　株式会社ソフトヴィジョン　126

プログラム「遊」工房　四乃巻

「マスターマインド」ゲーム　谷口典生　70

MS-DOSツール・ラボ 第1回

ディレクトリ表示ツール L.EXE 加野島英渡　76

実践C塾 第4回

コードヘルパと浮動小数点演算　水居 浄　84

プログラマのためのハードウェア再入門4

フロッピーディスク　鎌田正寿　94

98活用アセンブラ技法/ビデオシステム編 第4回

アセンブラと高級言語のリンク・および常駐サブルーチン 星野操　98

情報処理技術者試験 第2種合格！トレーニング

第1回 ハードウェアの基礎Ⅰ 勝島貴之　113

INFORMATION PAGE　129

NEWS & NEW PRODUCTS　130

Programmer's Program QuickHelp　134

オンラインソフト採集帳4「LHa Ver.2.05」136

HEALTHY? UNHEALTHY?

SE-NET 拾い読み　138

ドラゴン・ポスト　140

バーチャルスペースガイド　141

読者プレゼント

付録ディスクのご案内　142

クイズ・ランダムスクーン　143

オンラインサービス(SE-NET)のお知らせ

付録

5"2HDフロッピーディスク

綴じ込み付録

Turboシリーズ プログラム起動オプション

広告索引　資料請求番号

AX青樹会……P02

テックソフト＆サービス(株)……001

(株)ボーランドジャパン……002

日本システム(株)……004

丸文(株)……P03

三菱電機(株)……P04

人材募集広告

(株)エルテックス

(株)ソフトヴィジョン

(掲載順)

C Reader

特集
C読本

C 言語と正しく付き合うためのポイント

パーソナルコンピュータの短い歴史の中で，C 言語ほど広く，そして深く浸透したプログラミング言語は存在しない．C 言語は，プロからアマチュアまで，あらゆるレベルのプログラマから支持されている．UNIX というミニコンピュータ用のオペレーティングシステムを記述するために生まれた言語が，なぜこうもプログラマから好まれるのだろうか？　その質素な言語仕様とコンピュータのすべての資源にアクセスできる自由さに，古き良きハッカースピリッツを感じるからだろうか？

90 年代に入り，オブジェクト指向という新しいプログラミングパラダイムが普及しようとしているいま，C 言語はピュアなオブジェクト指向言語に遅れをとるどころか，この新しいパラダイムを自らの言語仕様に吸収することでさらなる前進を図ろうとしている．このタフで頼もしく，それでいてちょっぴり扱いのやっかいな言語ととことん付き合ってみようではないか．

C Reader

1

Cを知らないあなたのための
体で覚えるC言語入門

種村明宏

プログラミングの入門者にC言語を勧めるべきかどうかという点については，いろいろ異論もあると思われる．しかし，現在もっとも多く話題になるプログラミング言語がCであり，またもっとも多くの情報が公開されているのもCである．初心者が話題の言語に興味をもつのはごく自然なことであるし，またより多くの情報が得られるほうが学習しやすいことも確かである．
この章では，著者の体験をもとに，Cプログラミングの世界への入門法を伝授する．Cプログラミングを身につけるのに，文法書だけを読んでいたのではいつまでたってもプログラムを書けるようにはならない．まず人の書いたソースを読み，打ち込み，さらにそれを改造してみる．誰もが，このような体験を通してCプログラミングの特徴を知り，その初歩を身につけてきたのである．

はじめに

とにかくリストを打ち込もう！

ひところ全盛だったBASICはすっかり影を潜め，いまはCの花盛り．べつにCじゃなければいけないという理由もありませんが，本屋さんに行くとプログラミングの本はほとんどC．最近は，雑誌にもC専門のものが出てきたりで，一種のブームになっています．

その反面で，Cをやりたいけれど市販の本を読んでもわからない，Cはむずかしい！　といった悩みも出てきています．そこで今回は，はじめてCに触れる人をおもな対象に“Cと楽しくつき合う方法”を研究してみようと思います．

ここではあまり複雑な話はしません．とにかくプログラムを打ち込んで身体で理解してください．私の変な（？）文章を理解するよりもずっと効果的です．

さあ，Cプログラミングの世界へ！

本を読むだけでなくソースと取り組もう

Cプログラミングに入門した人が必ず読むCのバイブルに，K&R(B.W.カーニハン ＆ D.Mリッチー著『プログラミング言語C』，共立出版，著者の頭文字をとってこう呼ばれる）があります．しかし，いきなりK&Rを読むのはほかの言語（とくに手続き型の言語）でプログラミングの経験を積んできた人でもないかぎりむずかしいと思います．はじめから文字だらけのお堅い本を渡されて読んでも，頭に入らないでしょう．

K&Rも第2版では，プログラミングの初心者向けに親切に書かれていますが，それでもつらいものがあると思います．そこで，K&Rに入る前の入門書が必要になるわけです．Cに関する本はたくさんあり選ぶのはたいへんです．何十，何百冊とある書籍のなかから自分の目的にあったものを選ぶのは初心者にとって簡単ではありません．

私は，自分にとっておもしろそうな（あるいは使えそうな）ソースリストが出ている本を選ぶのがよいと思います．というのは，プログラミングに慣れるために，人の作ったソースリストを打ち込んでそれを動かしてみるという方法があるからです．ただ，あまりむずかしい本でなく，ごく短いソースがなるべく多く載っている本がお勧めです．

パソコンの場合，昔（といってもほんの7～8年前）は，現在のように市販のアプリケーションソフトウェアが豊富ではありませんでした．自分で作るか，雑誌に載ったソースリストを打ち込んで利用していました．とくに，私はゲームなどを中心に（当時はBASICとアセンブラでしたが）プログラムを打ち込んでいました．いま第一線でプログラマとして活躍している多くの方もこういった経験をされたと思います．

とりあえず打ち込むだけなら，誰にでもできると思います．打ち込んだソースリストをコンパイラでコンパイルするときには，MAKE，コンパイラ，リンカなどの使い方を知らなくてはなりません．ただし，統合化環境をもったCコンパイラが増えてきているので，以前ほどむずかしいものではなくなってきています．

さて，打ち込み終わったあとのプログラムですが，そのまますんなりと動く場合はめずらしく，ほとんどの場合，

▶ COLUMN 1

▶ as is

UNIXの世界には，「as is」と呼ばれるソフトウェアの配布習慣があります．配布するほうは，インストールの手順，全体の構成などを文書で説明するだけで，残りはソースコードが文書の代行をしてくれるという考えで，配布用のパッケージが作成されています．

したがって，使用者はその内容を自分で把握しなければなりません．しかし，配布されたソフトウェアに関する責任を配布元で取らないかわりに，配布先ではその内容を自由に変更できます．

私もいくつかUNIXのフリーソフトウェアを利用していますが，場合によっては自分のシステムに合わせるためにソースコードの変更が必要となり，苦労したこともあります．このおかげで，それまで知らなかった技術や知識を身につけることもできましたが．．．

このような配布形態をとるソフトウェアのソースコードは，とても見やすく整えられていて，コメントも豊富です（その逆のケースもありますが．．．）．自分がソフトウェアを作成するときの参考になります．

打ち込みミスやバグのために思うように動作してくれません．そこで，言語やマシンのマニュアルとにらめっこしながらソースリストをいじるわけです．ここでプログラムの書き方や言語の関数など，その言語の使い方を自然に覚えていくわけです．

また，正常に動いたあと当然のようにわき起こるのが，このプログラムを改造してみようというハッカー的好奇心です．現在，ハッカーという言葉は悪い意味で使われることが多いのですが，もともとはコンピュータをより深く知って使いこなそうという気持ちをもった人を意味する言葉です．なにかを学ぶときには，そうした前向きの気持ちが大事です．たとえば，このゲームをもっと使いやすくしてやろうとか，裏技を使えるようにしてしまおうとか，いろいろ考えられると思います．

ソースをいじるには，まずソースリストが読めなければなりません．最初はたいへんだと思います．まずは，自分にあっている大きさで，「これならやってみる価値はありそうだ」「私にもできそうだ」と思えるプログラムからはじめましょう．

ソースリストは，本や雑誌に載っているもののほかに，パソコン通信や友人からソース付きのフリーソフトウェアを入手するのもいいでしょう．ただし，フリーソフトなどを改造するときは，作者の著作権を侵害しないように十分注意する必要があります．勝手に第三者へ再配布はできません．

CはもともとUNIX上のプログラミング言語です．UNIXでは，プログラムがソースで配布されることが多く，「as is」という形をとっています（COLUMN 1参照）．自分の好きなようにソースを変更することができるため，UNIXプログラマの技術向上，異機種間におけるプログラムの移植に役だっています．パソコンでソースコードの配布を行うことが少ないのは残念なことです．

とにかく，どん欲にいろいろなソフトをコンパイルしてみることからはじめてみましょう．

Cって何？

システム環境がCを求めている

人と付き合いはじめるとき，なぜか気になってしまうのがその人の過去（私が詮索好きというわけではないですが），プログラミング言語も例外ではないでしょう．

コンピュータのプログラミング言語は，一概にどれがよいとはいえません．それぞれ特徴をもっており一長一短があります．Cは比較的万能ですが，科学技術計算や事務処理計算では歴史の古いFORTRANやCOBOLが現在でも主流です．推論など，人工知能的なデータ処理にはCよりもLISPのほうが使いやすいでしょう．

低レベルな高級言語

Cは，UNIXシステムの記述言語です．それまでアセンブリ言語で記述していたシステムをCで書くことによって，開発，保守，異機種への移植を容易にしたわけです．アセンブリ言語は機械語と1対1に対応していますから，プログラムは小さく実行速度も速くなります．しかし，アセンブリ言語はあまり生産性がよくありません．ソースリストは高級言語にくらべて読みにくく，デバッグも容易ではありません．Cはそのアセンブリ言語のかわりとして作られたので，アセンブリ言語ほどではありませんが，同じような特徴をもっています．この特徴は，Cが“低レベルな高級言語”といわれる所以にもなっています．

ほかの高級言語を使っている方は，同じアルゴリズムのプログラムを自分の使っている言語とCの両方で書いてみて，コンパイル後のコードをくらべてみるとおもしろいと思います．また，アセンブリ言語でプログラミングしてきた方も，Cプログラムを書いてアセンブリ言語のソースリストを出力してみてください．最近のCコンパイラがいかにすごいかわかると思います．

ただし，Pascalなどにくらべて自由度が高い反面，エラーチェックや変数・関数の型チェックが甘いので，「コンパイルがうまくいったのに，実行してみたら暴走してしまった」ということもよくおこります．しかし，コンピュータに暴走はつきものです．よほどのことをやらないかぎり（とくにファイルやディスクをいじくるようなこと），実害はないでしょうから，はじめのうちは暴走と戦ってください．暴走させるたびに原因をつきとめてプログラムを修正していけば，それがプログラミング技術を上達させることにもなります．本当は，メモリ保護や資源保護のきちんとしたシステム（とくにオペレーティングシステム）なら，暴走してもシステム全体が落ちるということは避けられるのですが……．

ANSI規格のCとは

最近，CはANSI規格で言語仕様が統一されました．これまでバラバラになっていたC言語の規格を標準化して

MS-Cエラーメッセージ
Fatal error C1000: UNKNOWN FATAL ERROR Contact Microsoft Product Support services　コンパイラに予期しなかったエラーが検出された．

移植性を高めるとともに，Cの欠点であった関数や変数の型チェックを厳密化するなど，仕様の拡張が行われています.「ANSIはやだ！」とダダをコネる人がいても大丈夫です．ANSIでは，従来のCの仕様でも通るようにできています．ANSI規格で書かれたCのソースリストは比較的楽に，どのコンピュータのCコンパイラ（ただしANSI準拠の）でもコンパイルできます．しかし，ANSI規格が確定したのがごく最近であり，完全には対応していないコンパイラがほとんどです．CのANSI対応はこれからますます進むと思いますので，やり方によっては移植性の高いソフトを構築できるようになるでしょう．UNIXでは，CPUもメーカーも違う機種で，システムにあった条件でコンパイルしなおすだけで動作させることができます．

MS-DOSは，OSとしての機能がたいへん貧弱だったため，MS-DOS上で開発されたプログラムのほとんどが機種依存性の高いものでした．WindowsやOS/2はMS-DOSで不足していた機能をほとんどもっているため，これらのシステム上で開発されたプログラムの多くは機種依存性が低く，再コンパイルするか，あるいは実行形式のままでOS/2やWindowsの動くほとんどの機種で動作します．こうしたシステム上の主力言語となっているのがCです．よい悪いは別として，現在汎用に使えるプログラミング言語としてCはベストだといえます．

▶ COLUMN 2

▶ Cの歴史

1963年　ケンブリッジ大学とロンドン大学の共同プロジェクトによりCPL（Combined Programming Language）を開発．CPLはAlgol60（1960年）の影響を受けている．

1967年　ケンブリッジ大学のMarth RichardsがBCPLを開発（Basic CPL）．

1970年　Ken Tompson，DEC PDP-7用の最初のUNIXシステムでBCPLをもとにB言語を開発（B言語のBはBCPLの頭文字であると同時に，彼の妻の名であるBourneの頭文字でもある．BCPLおよびBにはデータの型がなかった）．Ken Tompsonは，このあとBのローカルバージョンであるNB言語を開発，これがCの直接の祖先となった．

1973年　ベル研究所でDennis RitchieらによってC言語が開発される．

1978年　「プログラミング言語C」（K&R）の第一版が出版（日本版は1981年7月）される．

1983年　アメリカ国内標準協会（ANSI = Ameican National Standard Institute）は，Cに関する現代的で総括的な定義を行う目的の委員会を発足．

1988年10月31日　「アメリカ国内情報システム標準規格（ANSI）……プログラミング言語C，X3．159-1989」としての承認を求めてANSIに規格案を提出．

1988年末　ANSI標準規格ANSI C承認．

1989年　「プログラミング言語C第2版」出版（日米ほぼ同時）．

Cプログラミングの基礎

関数という名のマジックボックス

プログラムを書く前に

Cのソースリストの特徴をひとことでいえば，関数の塊ということになります．簡単にいえば，何かものを入れたらそれを変えてくれる，またはそれを使えば何かをしてくれるマジックボックスがいくつもあるというかんじです．Cのプログラムを書くことは，これらのマジックボックスの仕掛けを作り，うまく手品を演じることです．

Cが最初からもっている予約語は，ほかの言語にくらべてほんのわずかです．まず，これくらいは覚えてしまいましょう．極端なことをいえばこれがCのすべてです．あとはこれらを組み合わせてマジックボックスの仕掛けを書けばいいだけです．

しかし，これだけですべてのプログラムを書くのはたいへんです．そこで多くの便利な関数があらかじめ用意されています．それを集めたものを標準

ライブラリと呼んでいます．標準ライブラリに登録されている関数をすべて覚える必要はありません．まず，よく使う関数だけを覚えて，ライブラリのリファレンスを辞書のように使うといいでしょう．どの関数をよく使うかは，人それぞれだと思いますが，まずは他人のプログラムを解読してみて，それで覚えるのがよい方法だと思います．

できればハンディなリファレンスマニュアルがあるといいのですが，コンパイラに付属してくるたいていのCのマニュアルはぶ厚く，初心者は必要な情報を得るのに苦労します．このようなときは，市販のマニュアル本を利用します．Cの本には，オリジナルのマニュアルの必要な部分だけを簡潔にまとめたマニュアル本と，プログラミング本の2種類があります．マニュアル本はオリジナルよりも読みやすくハンディなため，もち歩くことができて便利です．最近では，オンラインマニュアルの付いているCコンパイラも多く，この場合はいちいち本をめくらなくても必要な情報を迅速に引き出せてたいへん便利です．

▶ COLUMN 3

▶ Cの予約語（ANSI）

予約語	機能
auto	動的記憶期間を指定する
break	switch文本体またはループの実行を終了する
case	switch文中でテーブルのラベルを示す
char	1バイトのデータ型を指定する
const	値が変更できないオブジェクトを指定する
continue	ループ本体の残りをスキップして次の繰り返しを開始する
default	switch文中でdefaultラベルを作成する
do	doループを構成する
double	倍精度浮動小数点数を示す
else	条件分岐して文を実行する
enum	データ型を列挙する
extern	外部変数とのリンクを指定する
float	単精度浮動小数点数を示す
for	条件付きループを構成する
goto	関数内での無条件分岐を行う
if	条件分岐の際に判断制御を行う
int	整数を示す
long	long int，signed longなどの型指定子を構成する
register	高速な動的記憶期間を指定する
return	呼び出した関数へ制御を戻す
short	short int，signed intなどの型指定子を構成する
signed	符号付きの整数を示す型指定子を構成する
sizeof	引数のサイズをバイト単位で返す演算子
static	識別子が内部リンケージをもつことを宣言する
struct	構造体指定子を構成する
switch	表の中からラベルを選択する
typedef	他の型の同義語を示す
union	共用体指定子を構成する
unsigned	符号なし整数を示す型指定子を構成する
void	空の型を示す
volatile	識別子が頻繁に変更されることを示す
while	whileループを構成する

プログラムを書いてみよう！

それではプログラムを書いてみましょう．リスト1を見てください．たいていどの本にも載っているいちばん短いプログラムです．ANSI，非ANSIに限らず，どのCコンパイラでもコンパイル可能なはずです．このプログラムは，マジックボックスの箱だけが用意されていて，仕掛けは何もありません．ですから，実行しても何も起こらないわけです．

Cのプログラムは {} によってブロックに分けられます．そのいちばん大きな単位が関数です．リスト1には，このブロックが1つだけあります．その中にさらにブロック（複文）を入れることができますが，残念なことにCではPascalのように関数の中に関数を定義することはできません．考えようによっては必要のないことかもしれませんが，Pascalを使ってきた方は不便に感じられるかもしれません．メインルーチン「main()」も関数です．引数はコマンドライン引数を取ることが

MS-Cエラーメッセージ
Fatal error C 1002: compiler is out of heap space in Pass 2　コンパイラがC 2.EXEの実行中に，ダイナミックメモリスペースを使い果たした．

でき，呼び出し元（環境）に対して値を返すこともできます．

フィルタ

言語にはそれぞれ特徴がありますが，Cらしいプログラムとはいったいどんなものでしょうか？ CはUNIX上で発達した言語ですから，UNIXの特徴を使ったプログラムはCの特徴を明確に表します．その1つがフィルタです．リスト2は非常に短く簡単なプログラムですが，これだけでもフィルタになります．I/Oリダイレクトとかパイプとかの概念は，もともとUNIXから発生したものです．フィルタはこの性質をうまく利用しています．

リスト2は，小文字を大文字にかえるフィルタです．コメントを除いた最初の2行はプリプロセッサに対する指示で，あとで使用する関数を宣言しているヘッダファイルをインクルードしています．stdio.hはgetchar()の，ctype.hはtoupper()を使うためにインクルードしています．どの関数がどのヘッダをインクルードするかは，マニュアルなどを参照してください．

その次のmainの前に付いているvoidは，この関数は値を返さないという型宣言です．この宣言をしてあるにも関わらず，return ・・・;とするとそこでエラーとなります．正確にはコンパイル時にWARNINGとなるだけですのでコンパイル自体は通ってしまいます．Cはほかの言語にくらべると，このあたりのエラーチェックが甘くできています．WARNINGが出てもしかたのないケースもあるのですが，たいていのCコンパイラにはWARNINGレベルの設定があって，あまりうるさいときにはエラーチェックを甘くすることができます．最初のうちは，WARNINGはすべてなくすように心がけましょう．

このプログラムでCらしい表現を使っているのはwhile文のところです．while文は()の中の条件が真の間{}のブロック（次の文，関数が1つの場合省略可）を実行し続けます．whileだけでなくほかの制御文にもいえますが，Cでは条件式をかなり自由に()の中に入れられます．これによってプログラムの記述を簡潔にすることができますが，あまりやりすぎるととても読みづらいソースになってしまいます．

リスト2では(c=getchar())!=EOFとなっています．これは，getcharによってcに取り出された値が，EOF (End of File) になるまでwhile文の中を実行します．この部分はフィルタの基本部分で，必ずといってよいほど出てきます．toupper()は，正確には関数ではなくマクロです．ctype.hの中でリスト3のように定義されています．

このプログラムを少し変更するだけ

▶ COLUMN 4

▶ I/Oリダイレクト&パイプ

UNIX，MS-DOS，OS/2などのOSでは，入出力資源はすべてファイルとして扱われます．通常は出力先・入力元のデバイス（ファイル）は決まっていますが，そのファイルを変えることで，ファイルから直接文字を読み込んだり，画面への出力をプリンタに出力・印字することができます．パイプは，水道管に水を通すようにデータをプログラムからプログラムへ流してやります．

MS-DOSの場合は，つねに1つのプログラムしか動くことができないため，送り側のプログラムのデータを一時テンポラリファイルにすべて流して，そこから受け側のプログラムへデータを流します．UNIXやOS/2では，プログラムを複数同時に動かすことができるため，その必要はなく，流されたデータは直ちに受け側のプログラムによって処理されます．本来のパイプの意味は後者のものです．

list

▶リスト1

```
main()
{
}
```

▶リスト2

```
/* Filter program */

#include <stdio.h>
#include <ctype.h>

void main()
{
        int c;
        while ((c=getchar())!=EOF)
                putchar(toupper(c));
}
```

MS-Cエラーメッセージ
Fatal error C 1003: error count exceeds n; stopping compilation　プログラム内のエラーが多すぎるか，重大すぎるため回復できない．

で，入力を出力にコピーするプログラムになります(リスト4)．たとえば，MS-DOSやOS/2では“FILTER2 < TEST.TXT > TEST2.TXT”とすることで，ファイルの複製を作成することができます．見てわかるとおり，本当はこちらのほうが基本形です．このプログラムは，たんにFILTER2 < TEST.TXTとすることで，UNIXのcatやMS-DOSのTYPEのようにテキストファイルを画面に表示するプログラムになります．また，標準出力先をプリンタに（DOSではPRN）することで，テキストファイルを打ち出すこともできます．このプログラムをちょっと細工したものがリスト2だったわけです．

次にこのプログラムを少し変更して，入力した文字列に行番号を付加するフィルタを作ってみます（リスト5）．これまでのフィルタは文字単位の処理でしたが，このフィルタでは行単位の処理を行っています．1行の最大文字数は256文字です．

while文の中を見てください．いままでgetcharにより文字単位で標準入力から取得していたのに代わって，fgetsを用いてリターンキーが押されるまでに入力された文字列を取得しています．ここでは，第3パラメータにstdinを指定して，標準入力からの入力にしています．

printfの中の“%04d”とか“%s”は出力の書式設定です．%04dは4桁の整数で，%sは文字列を出力します．ほかにもいろいろ設定がありますが，最初は自分でよく使うものだけを覚えておくといいでしょう．この書式設定は，ほかの言語のそれにくらべてめんどうですが，プログラムを書く上でたいへん便利なときがよくあります．たとえば，文字列や変数の中身を16進数で出力したいときなどにも変換作業が必要ありません．また，桁数を指定して出力したいときや，出力スタイルを指定したいときなどに便利です．

count++という表現ですが，これはcount = count+1;と同じ動作をします．++はアセンブラにおけるインクリメント命令と1対1に対応しており，後者の表現よりも高速にかつメモリを無駄に消費しません．同じような表現に--がありますが，こちらはデクリメント命令と対応しています．また，++countという表現もできますが，これは前置演算（それに対してcount++は後置演算という）といい，++count;とした場合は後置演算と同じですが，i=0としたとき，a[i++]=0;とa[++i]=0;とは意味が違います．前者はa[0]=0を実行後iをインクリメントして次のステップにいきます．後者の場合，a[1]=0を実行してから次のステップにいきます．

▶ COLUMN 5

▶プリプロセッサ&ヘッダ

Cでは，プログラム本体を記述する前に各種の定義，マクロ，ファイルのインクルードを行うことができます．

プリプロセッサで使われる疑似命令の一覧を示します．

疑似命令	機能
#define	マクロ定義を行う
#ifdef	定義済みの条件組み込みを指定する
#elif	#if，#ifdef，#ifndefの偽部での判定の始まりを示す
#else	#if，#ifdef，#ifndef，#elifの偽部の始まりを示す
#endif	#if，#ifdef，#ifndefで始まる前処理の終りを示す
#if	真偽の条件組み込みを指定する
#ifndef	未定義の条件組み込みを指定する
#include	指定したファイルを取り込む
#line	行番号をつける
#undef	#defineによるマクロ定義を無効にする
#pragma	処理系に依存した部分の処理を行う

ヘッダは，使用する関数，マクロ，各種定義，定数，グローバル変数などを宣言するところです．Cではプログラムを再コンパイルするときに，変更が必要な箇所をあらかじめヘッダファイルにまとめて宣言しておき，プログラム自体をなるべく変更しないようにするほうが懸命です．とくに機種依存情報などを，ヘッダファイル，Makefileにまとめておくことにより，上記の疑似命令を使って処理を分けることができます．

巨大なプログラムは，適当な長さのモジュールに分けて分割コンパイルするのが普通です．うまくプログラムをまとめておかないと，ごちゃごちゃになってしまいます．ヘッダファイルはこうしたモジュールのドキュメントにもなるので，なるべくコメントを詳しく書いて，モジュールの管理をしやすくしておくといいでしょう．

MS-Cエラーメッセージ
Fatal error C1004: unexpected end-of-file found　デフォルトのディスクドライブ上にテンポラリファイルを作るのに十分なスペースがない．

count+=2という表現もありますが，これもcount=count+2を略した表現です．正確には+=を代入演算子と呼んでいます．+に限らず二項演算子（+，-，*，/，%，<<，>>，&，^，|）も代入演算子をもっています．

注意すべきなのは，count *= a+bというのはcount=count *(a+b)の略になることです．これは，あまり長い表現をすると見にくいところでよく使われています．とくに右辺の式が長くなったときは通常どおり書くよりも見やすくなります．制御文の中でもよく使われています．すでにリスト2，4ではwhile文の括弧の中で“!=”として使っています．!は論理演算子notを表しています（ほかに|=or，&=and，^=xorがあります）．

代入文の使用は表現を簡略化するだけでなく，コンパイラがよりよいコードを生成するのにも役立ちます．

このプログラムをさらに少し変更すると，簡単なmoreが作れます（リスト6）．このプログラム中で使用しているgetchは，コンソールからの1文字直接入力です．同じような関数にgetcheがありますが，こちらは文字を画面にエコーバックする以外はgetchと同じです．

getch（getche）は標準的なCのライブラリには含まれていませんが，K&Rにも作り方が出ていますし，市販のCコンパイラにはたいていあります．MS-C，TURBO Cではインクルードファイルはconio.hになっています．

こうしたフィルタは機能を適当に分けて作成し，シェルスクリプト（バッチ）を利用して必要なものを組み合わせて使用すると便利です．フィルタには，ほかにもTAB←→SPACE変換とかUNIXではnkf（ネットワーク漢字コード変換），nroff，patchなどがあります．誰にも文章を読まれたくないときは，暗号化のフィルタをもっているといいですね．フリーソフトウェアとして流れている関西弁フィルタ（標準語→関西弁）もジョークソフトとしておもしろい存在です．どなたか東北弁フィルタとか，九州弁フィルタに挑戦してみてはどうですか？

list

▶リスト3

```
int _FAR_ _cdecl toupper(int);
int _FAR_ _cdecl _toupper(int);
#define toupper(_c)     ( (islower(_c)) ? _toupper(_c) : (_c) )
#define _toupper(_c)    ( (_c)-'a'+'A' )
```

▶リスト4

```
/* Copy program */

#include <stdio.h>

void main()
{
        int c;
        while ((c=getchar())!=EOF)
                putchar(c);
}
```

▶リスト5

```
/* Include Line Nummber */
#include <stdio.h>

void main()
{
        int count=1;
        char buffer[255];

        while (fgets(buffer,256,stdin)!=NULL)
                printf("%04d : %s",count++,buffer);
}
```

▶リスト6

```
/* More */
#include <stdio.h>
#include <conio.h>

#define maxline 23

void main()
{
        int count=1,c=0;
        char buffer[256];

        while (fgets(buffer,256,stdin)!=NULL){
                printf("%s",buffer);
                if (count==maxline){
                        printf("-- More --");
                        c=getch();
                        printf("¥n");
                        count=0;
                }
                count++;
        }
}
```

アルゴリズム

Cだけでなくプログラミング言語全般にいえることですが，プログラムを書く上でいちばん大切なのはアルゴリズムです．簡単にいえばプログラムの

流れなのですが，これが悪いと処理スピード，実行速度が遅くなるばかりか，バグも出やすくなります．簡単なストリングマッチングのプログラムを書いてみました(リスト7)．まずは打ち込んで動かしてみてください．

このプログラムは，先に入力した文字列の中にあとで入力した文字列が含まれていた場合に success となり，含まれていない場合に fail となり，入力を繰り返すプログラムです．このプログラム単体では非常につまらないものですが，いろいろ応用できそうです．たとえば，テキストファイルの中からの任意の文字列を検索するとか，言葉あてゲームとかです．

fig.

▶図1　ビットシフトの例

1110　を右に1ビットシフトすると　0111になる．12/2=7
0101　を左に1ビットシフトすると　1010になる．5*2=10

プログラムの最初の# define　cls　printf("¥33[2J")は，エスケープシーケンスです．¥33[2J は画面クリアのエスケープシーケンスで，ほかにもさまざまなエスケープシーケンスがあります．よく使うと思われるものは，まとめてライブラリにしておくと便利でしょう．エスケープシーケンスもANSIで規格が決まっていますから，対応していれば機種やシステムには依存しません．エスケープシーケンスについては，コード表を参考にしてください(表1)．このプログラムは，元のアルゴリズムとは少し変えてあります．whileループを forループで書き直したものですが，whileループを利

list

▶リスト7

```
/**********************************************************************/
/*                                                                    */
/*                        String  Matching                            */
/*                                                                    */
/**********************************************************************/

#include <stdio.h>
#include <string.h>

#define max 255
#define cls printf("¥33[2J")

void main()
{
        char a[max],b[max];
        int len1,len2,i=0,j=0,sflag=0;

        cls;
        printf("Please input stringth (max=255)¥n");scanf("%s",a);
        len1=strlen(a);

        while(sflag!=1){
                printf("Please Input Pattern(max=255)¥n");scanf("%s",b);
                len2=strlen(b);

                        for(i=0;i<=len1;i++){
                                if (a[i]==b[j]){
                                        j++;
                                        if (j==len2){
                                                sflag=1;
                                                break;
                                        }
                                }
                                else{
                                        if (j>0)
                                                i--;
                                        j=0;
                                }
                        }
                        if (sflag==0)
                                printf("Failure¥n");

        }

        printf("Success¥n");
}
```

▶リスト8

```
/* Quick Sort Program */
#include <stdio.h>
#define MAX 255
void Qsort(int *,int *);
void main()
{
        int n,i,a[MAX];
        printf("Input nummber N= ");
        scanf("%d",&n);
        for (i=0;i<n;i++){
                printf("a[%d] = ",i+1);
                scanf("%d",&a[i]);
        }
        Qsort(a,a+n-1);
        for (i=0;i<n;i++)
                printf("a[%d] = %d¥n",i,a[i]);
}
void Qsort (int *l,int *r)
{
        int *i=l,*j=r,x=*(l+(r-l>>1)),w;

        do{
                while ( *i < x)
                        i++;
                while ( *j > x)
                        j--;
                if(i > j)
                        break;
                w = *i;
                *i = *j;
                *j = w;
        } while (++i <= --j);

        if (l < j)
                Qsort(l,j);
        if (i < r)
                Qsort(i,r);
}
```


MS-Cエラーメッセージ
Fatal error C 1006: write error on compiler intermediate file　コンパイル作業に使う中間ファイルを作成できなかった．

用する元のアルゴリズムに直してみてください．どちらが最適かは自ら判断してみましょう．

次に示すのはクイックソートのプログラムです(リスト8)．プログラムを作るときには，よくソートのさまざまなアルゴリズムを使います．とくに，クイックソートのアルゴリズムでは再帰呼出（リカーシブコール）を使います．リカーシブコールを使うと，プログラムをシンプルにできますが，反面プログラムの流れが見にくくなります．また，スタックオーバーフローを起こしやすくなるので，大きな変数をグローバル変数にします（ここでは使っていません）．一般に再帰アルゴリズムは実行速度が落ちるので，なるべく再帰のネストを深くしないように心がけたほうがいいでしょう．

プログラム途中にある>>は，ビット演算の右シフトで，/2と同じ意味をもっています．つまり，アセンブラのshrに対応しているわけです．/2として割り算の演算ルーチンを呼び出すよりも，はるかに早く演算を行うことができます．逆に2倍するときは，左に<<でビットシフトします．

ちょっとした技ですが，これだけでも処理を高速化することができます．基本的なソートのアルゴリズムはこのほかにもいくつかあります．本で調べて，1つ1つ打ち込んで公式集のようにまとめておくといいでしょう．

ポインタ

Cを学ぶときに避けて通れないのがポインタです．Cはもともとアセンブラに近いものです．アセンブラは，ポインタ（メモリアドレス）を利用してデータ処理を行っています．通常の高級言語であまりポインタのようなものを見かけないのは，その大部分を隠してしまっているからです．C以外でもPascalではポインタをよく使いますが，Cよりも抽象化されていて，私は使いにくいような印象を受けました．

Cのポインタはどちらかというとア

fig.

▶表1　エスケープシーケンスコード表（PC-9801）

エスケープシーケンス	機能
ESC [pl;pc H	Direct cursor addressing
ESC [pl;pc f	ESC [pl;pc Hと同様
ESC =l c	ESC [pl;pc Hと同様
ESC [pn A	Cursor up
ESC [pn B	Cursor down
ESC [pn C	Cursor forward
ESC [pn D	Cursor backward
ESC [0 J	Clear from cursor to end screen
ESC [1 J	Clear from beginning of screen to cursor
ESC [2 J	Clear screen
ESC *	ESC [2 Jと同様
ESC [0 K	Clear from cursor to end of line
ESC [1 K	Clear from beginning of line to cursor
ESC [2 K	Clear entire line containing cursor
ESC [pn M	Delete line
ESC [pn L	Insert line
ESC D	Index
ESC E	Next line
ESC M	Reverse index
ESC [s	Save cursor position
ESC [u	Set cursor position
ESC [6 n	Cursor position report
ESC)0	Select kanji mode
ESC)3	Select graph mode
ESC [>5l	Enable cursor display
ESC [>5h	Disable cursor display
ESC [>1h	Enable bottom line
ESC [>1l	Disable bottom line
ESC [>3h	Select 20 line mode
ESC [>3n	Select 31 line mode
ESC [>3l	Select 25 line mode
ESC [ps;･･･;ps m	Character attribute
ESC [Pn;. . .;Pn p	Keyboard reassignment
ESC [" string" ;p	Keyboard reassignment
ESC [Pn;" string" ;Pn p	Keyboard reassignment

センブラのポインタに近いので，柔軟性があり，アドレス計算も楽です．Cでは関数に渡した引数の返り値を受け取るには，渡す変数のポインタを渡さなければなりません．そのために，たとえばscanfで値を変数に入力するにはその変数のポインタをscanfに渡しているのです．アセンブラと違うところは，アセンブラが機種による絶対アドレス指定（仮想化されている場合もありますが，それもCPUごとに違うことがほとんどです）に対し，Cではそれを隠しています（参照することはできます）．ですから，ソースを見るかぎりアドレスの違いを意識せずにプログラミングすることができます．

ただし，インテル系CPUのコンパイラでは64KB以上のデータを扱ったり，コードの大きさが64KB以上になったときはコンパイラでメモリモデルの指定を行ってコンパイルする必要があります．通常はスモールモデルのみを使用していればほとんど問題はないでしょう．MS-Cではスモールモデルやラージモデルでも，変数や関数の前にfarやnearをつけることによって，部分的にラージモデルやスモールモデルとしてコンパイルできます．

インテル8086系のCPUでは，16ビットレジスタでセグメントとオフセットの2つを指定することで，絶対アドレス20ビットを表しています．このため，Cプログラミングにおいてもセグメントの存在を無視できず，データ型はnearとfarに分けられています．関数の場合も，64KBを超える場合にはfarにしなければなりません．286プロテクトモードを使っているOS/2 Ver.1.XXでも同じで，80386プロテクトモードを本格的に使うOS/2 Ver.2.0にならないかぎりこの問題は解決しないようです．

データ構造

データ構造は，プログラミングの学習の最後につまずくところで，この問題に悩む人は多いようです．ここでは，参考書を紹介するだけにしておきます．K&Rと並んで評価の高い，Niklaus Wirthの書いた『アルゴリズム＋データ構造＝プログラム』，これはCではなくPascalの本ですが，データ構造，アルゴリズムの話が適切にまとめられています．Pascalを知らないと少々きついですが，中のプログラムをCに書き換える練習を繰り返せばかなりの知識が身につきます．最近では，Cによるデータ構造やアルゴリズムについての本が出ていますので，これらを利用するのもいいでしょう．

簡単に構造体の話をします．いままで出てきたデータ型はそれぞれ単体で（配列にしても）扱ってきましたが，これらのデータをひとまとめにして使用することができます．いくつかのデータを1つのデータブロックとして扱うわけです．

構造体はポインタと組み合わせることにより，リスト処理やツリー構造をもつことができます．リストは配列と違って大きさ（配列の最大数）をあらかじめ決める必要はなく，そのつどメモリを割り当てて動的に変化します．メモリの割り当てをできるかぎりデータ数を増やすことができます．逆に，データ数が少ないときは無駄にメモリを確保しません．ただし，ポインタの扱いに注意しないとプログラムが暴走してしまう危険性があります．

配列よりも柔軟性があり，ツリー構造をもてるため，より複雑なデータ処理を行うことができます．ここでは，いちばん簡単な線形リストのプログラムを作ってみます（リスト9）．

線形リスト（linear list）とは，ブロック（構造体）が次のブロックを指すポインタを1つだけ含んで連結されたブロック全体のことをいいます．ここでは，nodeという名の構造体を定義しています．中身はnoという数字の入る箱と＊nextという次の構造体へのポインタとなる（定義された時点では，まだ次の構造体へのアドレスは入っていません）変数が入っています．

この線形リストでは，2つの構造体へのポインタ変数を使います．＊pointは1つ前のブロックを指すアドレスをもっており，その時点で*topが指す構造体のnextに，そのアドレスを渡す役目をもっています．そのため，このプログラムでは先入れあと出

▶ COLUMN 6

▶プロトタイプ宣言

ANSI Cから，関数は使用する前にその型と引数の型を宣言することができるようになりました．これにより関数に対して誤った引数や型を使用している部分の発見が可能になり，デバッグがしやすくなりました．

また，未定義の関数もこれによりコンパイルの段階で発見できます．プロトタイプ宣言をあえてしない場合は，Pascalのようにメインルーチンの前に関数をもってくることも可能です．

MS-Cエラーメッセージ
Fatal error C 1008: no input file specified　コンパイルするファイルが与えられていない．

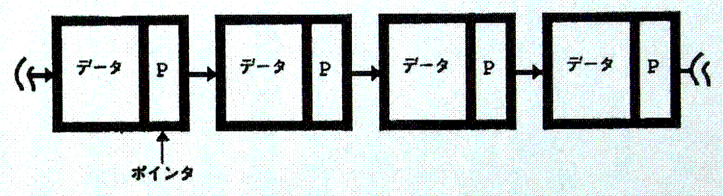
し (FILO) になっているわけです (実行すると，1，2，3と入力したのに対し3，2，1と出力されます).

＊topは，はじめに最初の構造体のアドレスをもっており，そのアドレスを＊pointに渡したあと，新たな構造体のためにメモリを確保します．メモリの確保はnew()関数で行います．中で使用しているmalloc()関数は，指定した大きさのメモリを確保して，そのアドレスを返す関数です．これを繰り返していくことにより，あと出しが可能な線形リストを作ることができます．プログラムを見ながら紙に図を描いて，追いかけてみるとわかりやすいと思います．このプログラムは，FILOになっているためデータの入力順序で出力できません．入力順に出力できるように工夫してみてください．

基本的なツリー構造やリスト構造は，公式のように形を覚えておくと便利です．Cのプログラミングでよくやることですが，このような部品プログラムをいくつも書いておいて，大きなプログラムを書くときにはそれらを組み合わせて利用するのです．

fig.

▶図2　線型リスト



OS サービスルーチンの利用

MS-Cからのシステムコールの利用方法を示します．DOSのサービスルーチンを使って，指定されたファイル名をディスクの中から取り出し，画面に表示するプログラムです．もちろんワイルドカードも使えます．ここではじめてコマンドライン引数を使っています．DOSのマニュアルを見て，この部分を解析してみてください．最初のargcは入力文字列の長さです．次の＊file_name[]は入力文字列へのポインタです．Cはメインも含めた全体が関数なので，コマンドライン引数もmain関数の引数となり，コマンドライン引数を実現するのも簡単です．

出力されるファイル名は，すべて配列変数の中に入っています．ソートのプログラムを追加することで，ファイ

list

▶リスト9

```
#include <stdio.h>
#include <malloc.h>
#include <conio.h>
#include <process.h>

/* データの構造体 */
struct node{
                                        int no;
                                        struct node *next;
} ;

/* Prottype */
struct node *new(void);
void Err(int );

void main()
{
        int a,c;
        struct node *top,*point;

        top = NULL;

        printf("Please input integer !\n");

        /* データの入力 & 線形リストの作成 */
        while ((scanf("%d",&a))>0){
                point = top;
                top = new();
                /* メモリわり当てできない場合Err()へ */
                if (top == NULL)
                        Err(1);
                top->no=a;
                top->next = point;
        }

        printf("Hit any key \n\n");
        c=getch();

        /* 入力データの出力 */
        for ( point = top;point != NULL;point = point->next)
                printf("%d\n",point->no);
}

/* メモリわり当て */
struct node *new(void)
{
        return((struct node *)malloc(sizeof(struct node)));
}

/* エラー処理 */
void Err(int ErrNo)
{
        switch (ErrNo){
                case 1 :printf("Not enough memory");
                                exit(1);
                                break;
                default:return;
                                break;
        }
}
```

ル名のサーチプログラムができます。ただし，配列変数の最大値がここでは64になっていますから，もっと多くのファイルを扱うときはこの値を大きくしてください。プログラムの最初にあるunionは共用体と呼ばれるものです。構造体と違って共用体のメンバは同じアドレスに配置されます。

リスト11のように，teはイメージ的には4つの型をもった変数となります。使い方は構造体と同じで，te.x，te→y，te.zとします。メンバのうち，もっとも大きいメモリを使うものに合わせてメモリが確保されます。

好奇心と遊び心をもって…試行錯誤は大切だが気分転換も忘れずに

かぎられたページで説明したことをいっぺんに理解するのは無理でしょう。ここでの話を参考に，よりよい学習方法を見つけるのがベストです。

紹介したプログラムのほとんどは不完全なものですが，これを改良してよりよいプログラムになおすことにトライしてみてください（手抜きのような…^_^;）。好奇心をもっていろいろ試行錯誤することが大切なのではないでしょうか。あまり堅く考えずに，いきづまったら，気分転換したほうがよいでしょう。

UNIXも最初はゲームを目的に生まれてきたといわれています。PC-9801もゲームなしではここまで普及したかどうかわかりません。コンピュータには遊び心をもってのぞむ，これが最高の学習方法だといまでも思っています。

list

▶リスト10

```
/*  This program is file name print FDIR.C              */
/*  programed by MS-C ver 5.1 /ver 6.0 on OS/2 v1.10 */
/*  for MS-DOS ver3.11/3.30                              */

#include <stdio.h>
#include <dos.h>

#define MAX 64

struct SREGS segregs;
union  REGS  inregs,outregs;

/* Prottype */

void set_DTA(int ,int );
int first_search(char *);
int next_search(void);
void error_msg(void);

/* Main routine */

void main(int argc,char *file_name[])
{
        char buffer[MAX];
        char file[MAX][13],*dummy;
        int i=0,j=2,k=0;

        /* 引数の大きさが0だった場合入力エラーメッセージを表示 */
        if (argc<=1){
                error_msg();
                return;
        }
        for (i=1;i<argc;i++){
                segread(&segregs);
                set_DTA(segregs.ds,(unsigned)buffer);
                if(first_search(file_name[i])){
                        dummy=buffer+0x1e;
                        while(*dummy!='¥0'){
                                file[i][k]=*dummy;
                                k++;
                                dummy++;
                        }
                }
                file[i][k]='¥0';
                k=0;
                while(next_search()){
                        dummy=buffer+0x1e;
                        while(*dummy!='¥0'){
                                file[j][k]=*dummy;
                                k++;
                                dummy++;
                        }
                                file[j][k]='¥0';
                                j++;
                                k=0;
                        }
                }
                for (i=1;i<j;i++)
                        printf("%s¥n",file[i]);
}

void set_DTA(int dataseg,int dataoff)
{
        inregs.h.ah=0x1a;
        segregs.ds=dataseg;
        inregs.x.dx=dataoff;
        intdosx(&inregs,&outregs,&segregs);
}

int first_search(char *fname)
{
        segread(&segregs);
        inregs.h.ah=0x4e;
        inregs.x.dx=(unsigned)fname;
        inregs.x.cx=0x06;
        intdosx(&inregs,&outregs,&segregs);
        return(!outregs.x.cflag);
}

int next_search(void)
{
        segread(&segregs);
        inregs.h.ah=0x4f;
        intdosx(&inregs,&outregs,&segregs);
        return(!outregs.x.cflag);
}

void error_msg(void)
{
        printf("Bad parameter !!¥n");
        printf("FDIR [PATH][FILE_NAME]");
}
```

▶リスト11

```
union  test{
                int   x;
                int   *y;
                float  z;
                char   *w;
}te;
```


MS-Cエラーメッセージ
Fatal error C 1010: compiler limit : macro expansion too big　マクロの展開が利用可能なスペースを超えた。

C Reader 2

ワークステーションからパソコンへ
UNIXのC，MS-DOSのC

辻村　浩

C言語がUNIXを記述するために生まれたことは，広く知られている．今でこそMS-DOSをオペレーティングシステムとするパソコン上で広く使用されているものの，C言語はもともとマルチユーザー・マルチタスクのUNIX上で使用されるべく設計されたプログラミング言語なのである．それに対して，MS-DOSはシングルユーザー・シングルタスクのオペレーティングシステムである．さらに，MS-DOSは80x86系のCPUを利用している関係から，セグメントという他のCPUには存在しない特異な概念をもっている．
これら，UNIXとMS-DOSという2つのオペレーティングシステム上のC言語の違いを明らかにするとともに，オペレーティングシステムの差を越えて，ポータビリティのあるソースを書く方法について詳説する．

C言語とUNIX

UNIXのために生まれたC言語

C言語は1972年にAT&Tのベル研究所で，DEC社のPDP-11用のUNIXを記述するために，Dennis M. Ritchieによって設計されました．UNIXはもともとアセンブラで記述されていましたが，機能拡張のしやすさ，保守のしやすさ，移植のしやすさを確保したいという要求から，高級言語で記述しなおすべきであり，そのための記述言語から設計すべきであるという考え方が起こってきたのです．そこで設計しようとした言語は，システムを記述する言語という観点から，機械に近いことが簡単に書けなくてはいけないし，移植性を確保するという観点からは，ある程度の可読性を備えた高級言語でなければなりませんでした．また，オペレーティングシステムの開発者が設計，開発を行う言語ということで，実際の開発に有用で効果のあるものでなければなりません．プログラミングの学習用言語とはわけが違うのです．

このように，UNIXを移植性のよいオペレーティングシステムにするためには，その記述言語が高級言語で，かつ同時に低級言語であるといった一見矛盾するともいえる命題を解決することが必要でした．そのような考え方のもとで生まれたのがC言語です．

しかし，いくら元となるオペレーティングシステムを高級言語で書き直したからといって，その記述言語が移植対象のマシンで動いていなければ，何の意味もありません．移植のしようがないですからね．そこで，UNIXの移植の第1段階として，言語処理系の移植が行われます．ということは，言語処理系自体の移植性がよくないと，この先の作業が滞ってしまうということです．そのため，C言語は処理系自体の実現のしやすさ，移植性のよさに重点が置かれて，改版されていきました．

UNIXは，DECのPDP-7という，主記憶や周辺装置の貧弱なミニコンに最初に載せられました．そのことが，UNIXやC言語がハードウェアに極力依存しないように設計され，拡張されていったことと無関係ではないと思われます．その甲斐あって，今ではUNIXとは直接関係のない，パーソナルコンピュータやメインフレームのシステム記述など，大規模なアプリケーションプログラムの開発用言語として，主要な言語処理系の一角を占めるまでに発展したのでした．

そして，UNIXがたゆまざる機能拡張と，最新の計算機科学の成果の取り込みを繰り返すなか，C言語は開発者の設計思想の表現手段として，あるいはデバッグの友として，しだいに洗練されていきました．その過程で，極力少ないキータッチでコーディングができるようにとか，わかりきったことを省略できるようにといった考えが取り入れられ，開発用の言語としては，この上なく魅力的なものに成長していくことになるのです．

そんなわけで，C言語はコンパイル時のエラーチェックが甘かったり，簡単に暴走するプログラムが書けてしまったり，安全性の面で問題があると長い間いわれ続けながらも，着々と普及していったというわけです．

MS-DOSとメモリモデル

64KBセグメントの壁

一方，MS-DOSはインテル社の8086というCPUのためのオペレーティングシステムとして開発されました．そのため，このCPUのもつ制限がオペレーティングシステム自体の制限，ひいては，その上で動作するアプリケーションプログラム自体の制限となっています．8086は64KBを1セグメントとするメモリ管理方式を採用しており，64KB以下のメモリ領域をアクセスする場合と，64KBを超えるメモリ領域にアクセスする場合とで，最適なアクセスの方法が違ってきます．

メモリのアクセスの方法はC言語にとっては重大な問題です．ポインタの概念を導入する上で避けては通れないからです．8086のメモリアドレッシングの問題を解決するには，アクセスする必要のある領域が64KB以下か，64KBを超えるかで場合分けして考えるのが簡単です．そこで登場したのがメモリモデルという考え方です．

コード領域，データ領域の双方について，64KB以下，64KB超という場合を考えると，図1の4つの場合があることになります．MS-Cではメモリモデルを，スモールモデル，コンパクトモデル，ミディアムモデル，ラージモデルと呼んでいます．ほかにもスモールモデルの特殊な形式としてのタイニィモデル（8ビットCPU用のオペレーティングシステムとして，かつて君臨していたCP/Mのロードモジュールに合わせた形式）や，ラージモデル

MS-Cエラーメッセージ
Fatal error C 1012: unmatched parenthesis nesting - missing 'character'　プロセッサ疑似命令内でカッコに対応するカッコがない．

の変形であるヒュージモデルも存在しますが，ここではこれ以上ふれません．要するに，MS-DOSを使う以上，メモリモデルの考えが頭に入っていないと，C言語を使いこなすことができないということなのです．

MS-DOSでは，メモリモデルに絡んで，farポインタとnearポインタという概念があります．すなわち，64KB以下の領域をアクセスするときに用いると効果的なのがnearポインタ，64KBを超える領域をアクセスするときに用いるのがfarポインタです．

MS-DOS用のC言語処理系にはポインタ型のデータの修飾子として，nearとfarが用意されています（最近はhugeというのもあります）．これは必ずしも使わなければいけないというものではありませんが，システムの資源の制限のきついプログラムを開発する場合に必要になってきます．

ファイルの概念

ファイルの終りはCtrl+Z?

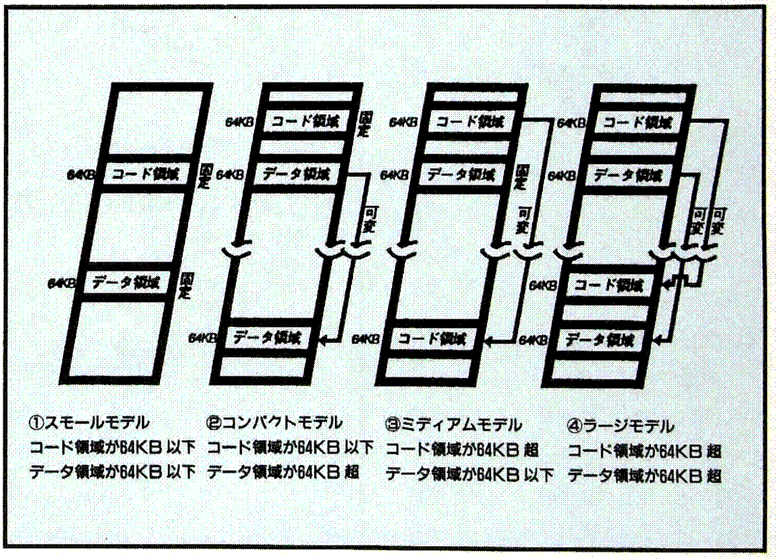
UNIXでは，ファイルはバイトのストリームであるという考え方を採用しています．つまり，ファイルは単なるバイト並びだという概念です．これに対し，MS-DOSでは，ファイルはテキストファイルとバイナリファイルの2種類に分けられ，前者は中身が文字列で構成され，後者はデータで構成されるという考え方を導入しています．

この違いはC言語上では，ファイルをオープンするときに現れます．つまり，UNIXだとファイルの中身に関係なく，入出力の引数だけを指定してopen()またはfopen()を使えばいいのに対して，MS-DOSでは対象とするファイルの中身や取り扱い方法に応じて，バイナリモードでオープンするかテキストモードでオープンするかを，さらに指定しないとなりません．

また，テキストファイルの終端を示すコード(EOFコード)も違っていて，MS-DOSでは0x1a (ctrl+Z) を用いていますが，UNIXではとくに終端であることを示すコードを入れる必要はありません．UNIXでは，キーボードからの入力の終りを示す場合は，通常0x04(ctrl+D)が用いられますが，これはUNIXの標準コマンドの1つであるsttyコマンド（端末設定表示・変更コマンド）を用いて，ほかの値に変更することもできます．もちろん，このコードはファイルの内容としては扱われません．

さらに，行の終り（改行）を表すコードも違っていて，UNIXでは0x0dだけでよいのが，MS-DOSでは0x0d，0x0aの2バイトが必要となります（UNIXもrawモードで使うときは，0x0d，0x0aになりますけどね．rawモードというのは入ってきたデータをいっさい加工しないで，つまり生(raw)のままの状態で次の工程に渡すモードのことです）．ですから，

▶図1　メモリモデル（MS-C）



▶図2　プロセス管理に関係するCの関数

機能	関数名	UNIX	MS-DOS
プロセスの複製	fork	○	×
プロセスの実行権の移動	execl	○	○
	execle	○	○
	execlp	○	○
	execv	○	○
	execve	○	○
	execvp	○	○
プロセスの実行	spawnl	×	○
	spawnle	×	○
	spawnlp	×	○
	spawnv	×	○
	spawnve	×	○
	spawnvp	×	○
子プロセスの終了待ち	wait	○	×
シェルエスケープ	system	○	○

MS-Cエラーメッセージ
Fatal error C 1014: too many include files　#include疑似命令のネスティングが限界の10を超えた．

UNIXとMS-DOSの両方で稼働するプログラムを開発するときや，UNIXのファイルをMS-DOS上に転送するときなどは，この違いに気をつける必要があります．困ったもんだ．

リスト2は，ファイルtest.data中のTABコード(0x09)ではじまる行数を求めるプログラムです．

そしてリスト3がMS-DOSの例で，ファイルtest.dat中に含まれる0xcd，0x1bのコード並びの数を数えるプログラムです．

プロセス管理

シングルタスクかマルチタスクか

UNIXがマルチユーザー・マルチタスクのオペレーティングシステムであるのに対し，MS-DOSはシングルユーザー・シングルタスクのオペレーティングシステムです．そのため，プロセスの管理の仕方が両者では異なります．UNIXでは，あるプロセスが自分の子供のプロセスを生んだあとも親子のプロセスが並行に動作することが可能なのに対し，MS-DOSでは子プロセスを生んだ親プロセスは子プロセスの実行が終わるまで実行権を放棄することになります．

プロセス管理に関係するCの関数には，図2のようなものがあります(MS-DOSはMS-Cの関数を示しました)．

実際に子プロセスを起動する例をリスト4に示しておきます．まず，UNIXの場合，子プロセス(/home/tuzimura/birthday)を実行している間，親プロセスが待っているという例

list

▶リスト1　far修飾子の使用例（MS-DOS）

```
const short HMAX = 80;
const short VMAX = 25;

main()
{
        short far *vram = ( short far *)0xa0000000L; /* VRAMアドレス */
        int       i;

        for ( i = 0; i < HMAX * VMAX; i++ )
                *vram++ = 0;
}
```

▶リスト2　UNIXのファイル入出力の例

```
#include <stdio.h>

main()
{
        FILE    *fp;
        char    *p, buf[ BUFSIZ ];
        int     l = 0;

        if ( ( fp = fopen( "test.data", "r" ) ) == NULL ) {
                fprintf( stderr, "cannot open test.data¥n" );
                exit( 1 );
        }

        while ( ( p = fgets( buf, BUFSIZ - 1, fp ) ) != NULL ) {
            if ( p[0] == '¥t' )
                l++;
        }

        fclose( fp );

        printf( "%d line(s)¥n", l );
}
```

▶リスト3　MS-DOSのファイル入出力の例

```
#include <stdio.h>

main()
{
        FILE    *fp;
        int     c, flag = 0, cnt = 0;

        if ( ( fp = fopen( "test.dat", "rb" ) ) == NULL ) {
                fprintf( stderr, "cannot open test.dat¥n" );
                exit( 1 );
        }

        while ( ( c = fgetc( fp ) ) != EOF ) {
                if ( !flag && ( c == 0xcd ) )
                        flag = 1;
                else {
                        if ( flag && ( c == 0x1b ) )
                                cnt++;
                        flag = 0;
                }
        }

        fclose( fp );

        printf( "count = %d¥n", cnt );
}
```

▶リスト4　子プロセスを起動する例（UNIX）

```
main()
{
        if ( fork() == 0 ) {        /* プロセスの複製 */
                /* 子プロセス */
                printf( "This is a CHILD process.¥n" );
                execl( "/home/tuzimura/birthday", "birthday", 0 );
        }
        else {
                /* 親プロセス */
                printf( "This is a PARENT process.¥n" );
                wait( 0 );          /* 子プロセスの終了待ち */
        }
}
```


MS-Cエラーメッセージ
Fatal error C 1016: #if[n]def expected an identifier　#ifdefと#ifndef疑似命令には識別子を指定する必要がある．

です．

次にMS-DOSですが，子プロセス(A:¥BIN¥BIRTHDAY.EXE)の起動を試み，成功したかどうかを表示するという例です．

list

▶リスト5　子プロセスを起動する例（MS-DOS）

```
#include <stdio.h>
#include <process.h>

main()
{
    int status;

    status = spawnl( P_WAIT, "A:¥¥BIN¥¥BIRTHDAY.EXE", "BIRTHDAY.EXE", 0 );
    if ( status == -1 )
        printf( "failed to execute child process¥n" );
    else if ( status > 0 )
        printf( "child process terminated¥n" );
    else
        printf( "Success!¥n" );
}
```

▶リスト6　パイプの例

```
main( argc, argv )
int     argc;
char    *argv[];
{
        int     p[2];

        if ( pipe( p ) == -1 ) {
            fprintf( stderr, "error!¥n" );
            exit( 1 );
        }
        if ( fork() == 0 ) {
                close( 0 ); dup( p[0] ); /* 標準入力をパイプにつなぐ */
                close( p[0] ); close( p[1] );
                execl( "/bin/grep", "grep", argv[2], 0 );
        }
        if ( fork() == 0 ) {
                close( 1 ); dup( p[1] ); /* 標準出力をパイプにつなぐ */
                close( p[0] ); close( p[1] );
                execl( "/bin/sed", "sed", "'/^$/'", argv[1], 0 );
        }
        close( p[0] ); close( p[1] );
        wait(0);
        wait(0);
}
```

▶リスト7　シグナルの例

```
#include <stdio.h>
#include <signal.h>

breakintr()
{
        printf( "BREAK!!¥n" );
        exit( 0 );
}

main()
{
        signal( SIGINT, breakintr );
        for (;;)
                printf( "now in loop¥n" );
}
```

プロセス間通信

MS-DOSには……ない！

UNIXにはプロセス間通信を実現するための手段として，パイプ(pipe)，シグナル(signal)，セマフォ(semaphore)，共有メモリ(shared memory)などが用意されています．

パイプ

パイプというのは，あるプロセスがパイプの一方の端にデータを流し込み，ほかのプロセスがパイプのもう一方の端から出てくるデータを受け取るという手法のことです．

パイプを使うには，まずパイプを生成しておけば，あとはパイプハンドルをファイルハンドルのように用いるだけでいいのです．このパイプハンドルをread()，write()で用いればデータの伝達ができます．

リスト6に示す例は，

sed '/$' ファイル | grepパターン

というコマンドラインを実行するのと等価なことをするプログラムです．

シグナル

シグナルは，ある事象（イベント）が起こったことをカーネルがプロセスに通知したり，プロセス同士で通知し合ったりする通信方法で，事象駆動(イベントドリブン）型のプログラムを書いたりするのに利用します．

シグナルで通知することのできる事象は，セグメンテーション違反（割り当てられていない領域をアクセスしようとした)，バスエラーやハングアップなどです．

例として，シグナルを使ったBREAKキー押下検出プログラムをリスト7に示します．

セマフォ

セマフォというのは，プロセスが現在，ある資源を占有中であることを，

ほかのプロセスに教えるための手段です．UNIX System Vでは，セマフォ関連のシステムコールを使って1つのプロセスにだけ資源のアクセス権を与え，ほかのプロセスの資源獲得要求をブロックすることができるようになっています．このような制御を排他制御といいます．

共有メモリ

共有メモリは，複数のプロセスが同時に使用できるメモリのことです．複数のプロセスのアドレス領域に同じメモリの場所が現れることになります．共有メモリの獲得には，shmget()システムコールを用い，この制御には先ほどのセマフォ・システムコールを使用してプログラミングをします．

MS-DOS

MS-DOSには疑似的なパイプが用意されているだけです．また，最近のMS-DOS用のC言語処理系は，BREAKキーが押されたことや数値演算プロセッサでエラーが発生したことをプログラムに知らせる機能をシグナルという名前で実装しています．

図3にプロセス間通信関係のCの関数を示します(セマフォ，共有メモリ，メッセージはUNIXのSystemVのものです)．

list

▶リスト8　セマフォの例

```
semget(セマフォ1つのセット);
while (他のプロセスのファイルアクセスの解除待ち) {
        semop(-1);          /* 共有メモリのロック */
        ファイルの使用
        semop(1);           /* 共有メモリのロックの解除 */
}
```

▶リスト9　共有メモリの例

```
/* プログラム1 */
semget(セマフォ1つのセット);
semop(1);              /* 共有メモリのロックの解除 */
while (他のプロセスのファイルアクセスの解除待ち) {
        semop(-2);          /* 共有メモリのロック */
        共有メモリの使用
        semop(1);           /* 共有メモリのロックの解除 */
}

/* プログラム2 */
semget(セマフォ1つのセット);
while (他のプロセスのファイルアクセスの解除待ち) {
        semop(-1);          /* 共有メモリのロック */
        共有メモリの使用
        semop(2);           /* 共有メモリのロックの解除 */
}
```

▶図3　プロセス間通信関係のCの関数

機能	関数名	UNIX	MS-DOS
FIFOパイプの生成	pipe	○	○
シグナル割込み処理の設定	signal	○	○
セマフォの制御	semctl	○	×
セマフォの作成	semget	○	×
セマフォの操作	semop	○	×
共有メモリの制御	shmctl	○	×
共有メモリの作成	shmget	○	×
共有メモリの操作	shmop	○	×
メッセージの制御	msgctl	○	×
メッセージの作成	msgget	○	×
メッセージの送信	msgsnd	○	×
メッセージの受信	msgrcv	○	×

システムコールのインターフェース

関数呼び出しとソフトウェア割り込み

UNIXでは，すべてのシステムコールはCの関数呼び出しで利用できるようになっています．MS-DOSでも，最近はCの関数呼び出しの形で用意されていますが，intdos()，intdosx()などの関数を使って，直接システムコール番号を指定して呼び出すこともできます．

MS-DOSでは，システムコールをソフトウェア割り込みという形で呼び出すことが可能です．

MS-Cエラーメッセージ
Fatal error C 1018: unexpected '#elif'　#if, #ifdef, #ifndef構造体の外で#elif疑似命令を使うことはできない．

入出力

入出力に関しては未定義

C言語は入出力の方法について，言語処理系としては定義されていません．一般に入出力(I/O)は関数呼び出しの形で実装されています．ただし，ioctl()に関しては，UNIXのいろいろなバージョン間で，またMS-DOSでも互換性がありません．

入出力制御は，MS-DOSではファンクションコールとして提供されていますので，C言語では低水準入出力関数，および標準入出力関数のライブラリとして利用することができます．さらにコンソール入出力関数や，8086系のCPUでは，I/Oポート入出力関数もあるのが普通です．

これに対してUNIXでは，低水準入出力関数，および標準入出力関数に関しては同じですが，コンソールやポートに対する制御はオペレーティングシステムの管理下にありますので，一般アプリケーションが直接制御することはできません．実際にはリアルタイムキー入力を行う場合，termio構造体を操作することになるのですが，少し話が込み入っていますので，ここでは触れないことにします．

まとめ

移植性より実行速度?

MS-DOS上の最新のC言語処理系に付いてくる標準関数ライブラリの充実には，目を見張るものがあります．そのため，C言語プログラムの数はしだいに増加しています．いまでは，MS-DOS上で使われているコンパイラ言語のほとんどがC言語になってきているといわれています．

また，今までUNIXでプログラムを組んでいた人がMS-DOSでもプログラムを組んだり，MS-DOS用のプログラムをUNIX上で開発したりする機会も増えています．しかし，今まで見てきたように，MS-DOSとUNIXでは同じC言語処理系とはいえかなり違いがあります．UNIXのプロセス間通信機能を利用したプログラムを，MS-DOSに移植するのはほとんど不可能ですし，MS-DOS固有のシステムコール呼び出しを用いたプログラムも，そのままでは移植できません．

結局，移植性のよいプログラムを書くには，オペレーティングシステムに依存した機能やハードウェアを制御する関数を用いてはいけないことになります．しかし，場合によっては移植性よりもプログラムの実行速度，ユーザーインターフェースの優しさ・見栄えのよさなどが優先されることがあります．とくにMS-DOSの場合は，ハードウェアを直接いじることが可能なので，いわゆる「行儀の悪いプログラム」が氾濫しているというのが現実です．

おまけ

whicth コマンドの紹介

最後に，UNIXでもMS-DOSでも動くプログラムの例として，私の作ったプログラムを紹介しましょう．このプログラムは，現在自分がどのオペレーティングシステム上で走っているかを確かめるために，環境変数pathの値を取得し，解析するという方法を取っています．あまりスマートではありませんが，こういう方法もあるのだということを知っていただければ幸いです（このプログラムはASCII-NET pcsの pool msdosにも登録してありますので，ご利用ください）．

list

▶リスト10　intdos()によるシステムコールの呼び出し例（現在時刻の表示）(MS-DOS)

```
#include <stdio.h>
#include <dos.h>

main()
{
        union REGS inregs, outregs;

        inregs.ah = 0x2c; /* 時刻の取得 */

        ret = intdos( &inregs, &outregs );

        printf( "02%d:02%d:02%d.02%d¥n", outregs.ch, outregs.cl,
                                         outregs.dh, outregs.dl );
}
```

whichコマンドの説明

書式

```
which <command> ...
```

機能

コマンドを実行しようとしたとき，実際に起動されるパス名を表示します．同じ名前でパスが違うコマンドがある場合（たとえば/usr/bin/mailと/usr/ucb/mailのような場合），設定されている起動パス検索順位により実際に起動されるプログラムが違ってきます．あらかじめ，本コマンドwhichで実際に起動されるプログラムはどれか，確認しておくことが可能です．

使用例

```
which ls
```

lsという名前のコマンド（MS-DOSの場合はls.com, ls.exe, ls.batの順で）の起動パスを調べます．

バグ

本コマンドのソースファイルはUNIX上およびMS-DOS上でコンパイルできます．ただし，UNIX上で/usr/bin/cshを利用している場合は，shell変数pathと環境変数PATHの設定内容にくい違いが生じることがあり，本コマンドが正しい起動パスを表示しないことがあります．これは現在，起動パスを検索するときに，関数getenv()で取得した環境変数PATHの値を利用しているためです．shell変数pathの値を求めるなんらかの方法を開発してもいいのですが，/usr/ucb/whichを使うほうが結果を早期に入手できるでしょう．したがって，本コマンドは/usr/bin/csh以外のシェル（たとえば/usr/bin/sh）の利用者向けということができるでしょう．もっともAT&Tのkorn shellやGNUのborn again shell(bash)上で本コマンドが動作するかどうかは確認していません．少なくともMS-DOSのcommand.comの利用者には使い道があるでしょう．

また，/usr/ucb/whichのように~/.cshrcを見て，aliasされているコマンドかどうかを調べることまでは行っていません．当然，~/.profileや~/.bashrcを見に行くこともありません．

作者

ANEWプロジェクトチーム　プログラミング担当（Anew is Not Equal Wwiv）
辻村 浩（ASCII PCS pcs30332, TSUPO-NET Tsupor-one #1）

TSUPO-NET 052-752-8053 300/1200/2400/9600bps MNP5/HST 24hours

履歴

Revision 1.6

MS-DOS Ver.3.xでappendコマンドでデータファイルの検索パスを指定している場合，そこで指定されているディレクトリに目的のコマンド（と同名のもの）があれば，あたかもカレントディレクトリに目的のコマンドがあるかのように表示される，という不具合を直しました．バグレポートをくださったYUTAKA氏(pcs 29035)に感謝します．

また，環境変数PATHで指定されているディレクトリのうち，いちばん最後のディレクトリを検索にいかないというバグの指摘をしてくださったMIA氏(TSUPO-NET #24)にも感謝します．まさか，こんな単純なバグが残っていたとは気が付きませんでした．（1990年9月2日）

Revision 1.5

一部分手直しを行うことにより，富士通のUTSでの動作を確認できました．MIA氏(TSUPO-NET #24)のご協力に感謝します．（1990年9月1日）

Revision 1.3, Revision 1.4

MS-DOS上で用いる場合，

```
which ls.bat
```

のように探したいコマンドに拡張子を付けた場合，MS-DOSの標準的なコマンド起動規則にしたがい，指定された拡張子を無視して，

```
ls.com
ls.exe
ls.bat
```

の順に探していくようにしました（今までは拡張子を指定するとnot foundになっていました．拡張子の指定をしない場合のみ，正常に動いていたわけです）．今まで通り拡張子を指定しなくても動作します．

また，富士通のUNIXすなわちUTSでは環境変数PATHに格納されている文字列が':'で始まらないためうまく動作しないとの報告がありました．さっそく，対応してみましたが動作は未確認です．（1990年8月31日）

Revision 1.1, Revision 1.2

暇つぶしにつくったコマンドですが，MS-DOS上で動くこの手のツールはまだほとんど見かけないので，公開してみることにしました．MS-DOS(COMMAND.COM)上とSUN-OS(/usr/bin/cshおよび/usr/bin/sh)上で動作確認をしましたが，マニュアルにもある通りcshを使う場合は実際の起動パスと合わないことがありますので注意してください．（1990年8月19日）

```
/*
 *  which.c
 *      check WHICH command will be executed when you type command line
 *
 *          by H.Tsujimura  Aug 19 1990
 *                 ASCII PCS    pcs30332  (Tsuporone)
 *
 *      Copyright 1990 by H.Tsujimura
 *      All Rights Reserved.
 *
 *      BUGS:
 *          When you hack on '/usr/ucb/csh', use '/usr/ucb/which'
 *          instead of this 'which'
 * History :
 * $Log: RCS/which.cv $
 * Revision 1.6  90/09/01 15:18:44  tsujimura
 * BUG FIX: MSDOSMODE --- (1) if command "APPEND" is used, "WHICH" prints
 *                            target exists current directory. (DOS3.x)
 *                             Then, existFile() is changed.
 *                        (2) "WHICH" doesn't search last directory of "PATH".
 *                             Then, added strcat( path, ";" );
 *
 * Revision 1.5  90/09/01 01:42:34  tsujimura
 * verified working on UTS(FUJITSU).
 *
 * Revision 1.4  90/08/31 09:31:58  tsujimura
 * BUG FIX: If <command> is at current directory, "which" ignores current
 *          directory. This bug born in Revision 1.3
 *
 * Revision 1.3  90/08/31 09:02:02  tsujimura
 * (1) MSDOSMODE: ignore any extention specified at command line
 * (2) UNIXMODE: support for environments variable "PATH" which doesn't begin
 *               with ':' (ex. FUJITSU UTS)
 *
 * Revision 1.2  90/08/19 07:36:32  tsujimura
 * searching current directory first (MS-DOS)
 *
 * Revision 1.1  90/08/19 06:58:26  tsujimura
 * Initial revision
 *
 *
 */

#ifndef lint
static char *rcs_id =
    "$Header: which.cv  1.6  90/09/01 15:18:44  tsujimura  Exp $";
#endif

#include <stdio.h>

#define MAXPATHLEN      255
#define MAXCMDLEN       160
#define MAXMSGLEN       512

#define MSDOSMODE       0
#define UNIXMODE        1
#define UNIXCSHMODE     2

int     mode = MSDOSMODE;
char    sep;

int
existFile( filename )
char    *filename;
{
        return ( !access( filename, 0 ) );
}
int
existCommand( cmd )
char    *cmd;
{
        static char     command[MAXCMDLEN];
        int             ret = 0;

    if ( mode == MSDOSMODE ) {
        char    *p = &cmd[1];

        while ( *p ) {
            if ( *p == '.' ) {
                *p = NULL;
                break;
            }
            p++;
        }
    }

        if ( mode == MSDOSMODE ) {
            char    *p = command;

                strcpy( command, cmd );
                strcat( command, ".com" );
                ret = existFile( command );

                if ( !ret ) {
                        strcpy( command, cmd );
                        strcat( command, ".exe" );
                        ret = existFile( command );
                }

                if ( !ret ) {
                        strcpy( command, cmd );
                        strcat( command, ".bat" );
                        ret = existFile( command );
                }

        while ( *p ) {
            if ( *p == '¥¥' )
                *p = '/';
            if ( ( *p >= 'A' ) && ( *p <= 'Z' ) )
                *p += 'a' - 'A';
            p++;
        }
        }
        else {
                ret = existFile( cmd );
        }

        if ( ret )
                printf( "%s¥n", mode == MSDOSMODE ? command : cmd );

        return ( ret );
}


char    *
getpath( path, num )
char    *path;
int     num;
{
        int             i = 0;
        char            *p;
        static char     str[MAXCMDLEN];

        strcpy( str, path );
        p = str;
        num++;
        while ( *p ) {
                if ( *p == sep ) {
                        i++;
                        if ( num == i ) {
                                *p = NULL;
                                break;
                        }
                        else {
                                strcpy( str, p + 1 );
                                p = str;
                                p--;
                        }
                }
                p++;
        }

        return ( str );
}


char    *
notExistCmdMsg( path, arg )
char    *path;
char    *arg;
{
        static char     msg[MAXMSGLEN];
        char            *p;

        sprintf( msg, "no %s in ", arg );

        p = &msg[ strlen( msg ) ];
        while ( *path ) {
                if ( *path != sep ) {
                    *p = *path;
                    if ( mode == MSDOSMODE ) {
                        if ( *p == '¥¥' )
                            *p = '/';
                        if ( ( *p >= 'A' ) && ( *p <= 'Z' ) )
                            *p += 'a' - 'A';
                    }
                }
                else
                        *p = ' ';
                p++;
                path++;
        }

        return ( msg );
}


void
main( argc, argv )
int     argc;
char    *argv[];
{
        int     i, j, found;
        char    *p, path[MAXPATHLEN], cmd[MAXCMDLEN];
        int     num = 1;
        char    *getenv(), *strchr();

        if ( argc <= 1 )
                exit( 1 );

        mode = MSDOSMODE;

        if ( ( p = getenv( "path" ) ) != NULL ) {
                if ( *p == '(' ) {
                        mode = UNIXCSHMODE;
                        sep  = ' ';
                        strcpy( path, p + 1 );
                        path[ strlen( path ) - 1 ] = NULL;
                        while ( *p ) {
                                if ( *p++ == sep )
                                        num++;
                        }
                }
        }
        if ( mode != UNIXCSHMODE ) {
                if ( ( p = getenv( "PATH" ) ) != NULL ) {
                    path[ 0 ] = NULL;
                    if ( ( *p != ':' )          &&
                         ( !strchr( p, ';' ) ) &&
                         ( strchr( p, ':' ) )      ) { /* FUJITSU UTS */
                        mode = UNIXMODE;
                        sep = ':';
                    }
                    else {
                            mode = ( *p == ':' ) ? UNIXMODE : MSDOSMODE;
                            sep  = ( mode == MSDOSMODE ) ? ';' : ':';
                            if ( mode == UNIXMODE )
                                    p++;
```

```
                        else
                            strcpy( path, ".;" );
                    }
                    strcat( path, p );
                    while ( *p ) {
                            if ( *p++ == sep )
                                    num++;
                    }
            }
            else {
                char    tmp[MAXPATHLEN];

                    mode = MSDOSMODE;
                    strcpy( tmp, ".;" );
                    strcat( tmp, path );
                    strcpy( path, tmp );
            }
            if ( ( mode == MSDOSMODE )                  &&
                 ( path[ strlen( path ) - 1 ] != sep )    ) {
                    strcat( path, ";" );
                    num++;
            }
    }

    for ( i = 1; i < argc; i++ ) {
            found = 0;
            for ( j = 0; j < num; j++ ) {
                    strcpy( cmd, getpath( path, j ) );
                    if ( mode == MSDOSMODE ) {
                        if ( cmd[ strlen( cmd ) - 1 ] != '¥¥' )
                            strcat( cmd, "¥¥" );
                    }
                    else
                            strcat( cmd, "/" );
                    strcat( cmd, argv[ i ] );
                    if ( existCommand( cmd ) ) {
                            found = 1;
                            break;
                    }
            }
            if ( !found )
                    printf( "%s¥n", notExistCmdMsg(path,argv[i]) );
    }

    exit( 0 );
}
```

C処理系紹介

■コンパイラ

●Turbo C

製品名：Turbo C 2.0／価格：19,500円／対応機種：NEC PC-9800シリーズ，IBM PC／問い合わせ先：(株)ボーランドジャパン　03-3486-1400

●MS-C

製品名：Microsoft C Professional Development System Ver.6.0／価格：98,000円／対応機種：PC-9800シリーズ，富士通FMRシリーズ／問い合わせ先：マイクロソフト(株)　03-3363-1201

●MS-QuickC

製品名：Microsoft QuickC Compiler Ver.2.0／価格：20,000円／対応機種：NEC PC-9800シリーズ，富士通FMRシリーズ／問い合わせ先：マイクロソフト(株)　03-3363-1201

●Lattice C

製品名：Lattice C/DOS Ver.4.1／価格：98,000円
製品名：Lattice C/DUAL Ver.4.1／価格：128,000円
製品名：Lattice C/ROMable Ver.4.1)／価格：128,000円
製品名：Lattice C/RT Ver.4.3／価格：238,000円
製品名：Lattice C V86/RT Ver.4.3／価格：298,000円
対応機種：NEC PC-9800シリーズ，富士通FM Rシリーズ，松下Panacom Mシリーズ，沖if800RX/EX，IBM PS/55シリーズ，東芝J-3100シリーズ／問い合わせ先：㈱ライフボート　03-3293-4711

●LSI C-86

製品名：LSI C-86／価格：48,000円／問い合わせ先：エル・エス・アイジャパン㈱　03-3404-1341

●Whitesmith C

製品名：Whitesmith,Ltd. C 86 コンパイラ Ver.3.25／価格：500,000円／問い合わせ先：(株)アドバンスド・データ・コントロールズ　03-3576-5351

●NDP C 386

製品名：NDP C 386／価格：148,000円／対応機種：NEC PC-9801シリーズの32bit機（ノーマルモードのみ），IBM PCとその互換機／問い合わせ先：テックソフトアンドサービス(株)　0474-23-1322

●HIGH C386

製品名：MetaWare HIGH C386／価格：148,000円／対応機種：NEC PC-9800シリーズの32ビット機，EPSON PC-386シリーズ，東芝J-3100シリーズの32ビット機およびJ-3300，IBM PC/AT互換機(80386，80386SX搭載機)／問い合わせ先：テックソフトアンドサービス(株)　0474-23-1322

●Power C

製品名：Power C compiler Ver.2.0.0J PC98版／価格：14,000円／対応機種：NEC PC-9800シリーズ，EPSON PC-286シリーズ／問い合わせ先：システム・ワン(株)　03-3663-4551

●Let's C

製品名：Let's C＋CSD／価格：25,000円／対応機種：NEC PC-9800シリーズ，各種MS-DOSマシン／問い合わせ先：パーソナルメディア(株)　パッケージソフトウェア営業部　03-3490-8841

■インタプリタ

●Advanced RUN/C

製品名：Advanced RUN/C／価格：29,800円／対応機種：NEC PC-9801シリーズ，富士通FM R50/60/70，FM TOWNS，東芝J-3100シリーズ，IBM PS/55シリーズ／問い合わせ先：㈱ライフボート　03-3293-4711

●C-terp

製品名：C-terp MS-C用／価格：39,800円／対応機種：NEC PC-9800シリーズ，IBM PC／問い合わせ先：(株)サザンパシフィック　045-314-9514

●Visual C

製品名：C統合化システムVisual C Ver.J1.01／価格：12,800円／対応機種：NEC PC-9800シリーズ（LT/XAおよびハイレゾモードでの使用を除く）／問い合わせ先：(有)データワン　0472-41-5641

●ORCHID C

製品名：ORCHID C／価格：20,000円／対応機種：NEC PC-9800シリーズ／問い合わせ先：㈱オーキッド　03-3827-7496

MS-Cエラーメッセージ
Fatal error C 1022: expected '#endif'　#if,#ifdef,#ifndef疑似命令の終わりに#endifがない。

C Reader

3

読みやすいプログラムとは？

Cプログラミングのスタイルを決める

秋山　登

構造化プログラミングの普及とともに，他人に読みやすい，人に優しいプログラム記述を心がけるという考え方も一般的なものになってきた．ほかの人が読んでもわかりやすいソースを書くことで，プログラムの保守がやりやすくなり，またプログラムの信頼性も高くなる．しかし，たんに構造化しただけでプログラムが読みやすく，信頼性の高いものになるわけではない．C言語は，構造化プログラミングを実現する構造化言語だが，その記述方法はプログラマの意志ひとつでどうにでも変えられる．インデントのつけ方ひとつ，{}のつけ方ひとつでCプログラムの読みやすさは，格段に変わってくるのである．ここでは，真に読みやすいソースとはどんなものか，Cプログラミングに求められるスタイルとはなにかについて考えてみたい．

プログラミング言語とは…

人間が読む言語でもある

プログラミング言語とは，人間が機械に行わせたいことを機械に伝える手段であるという認識をもっている読者は多いと思います．これは正しい認識です．しかし，現状のプログラミング言語の役割はこれだけではありません．機械の側から人間へ，プログラムについて的確な分脈に依存した情報が得られるほど，ハードウェアやソフトウェアの技術が十分に進歩していない現状では，プログラミング言語はそれを再利用したり，修正したりする人間（プログラムを書いた人も含めて）が読むための，人間同士のコミュニケーションの手段であることも認識しておかなければなりません．

プログラミングにおけるスタイルというのは，機械を効率よく働かせるという側面よりは，プログラミング言語を通じて人間同士のコミュニケーションを，いかに効率よく行うかを焦点とするものです．もちろん，このコミュニケーションという言葉にはプログラムを書いている人自身が自分のコーディングを良好に把握でき，バグの発生を未然に防止できるという意味も含まれます．

ところでスタイルという言葉は，生活のスタイルというような使い方をされるように，かなり個性や社会環境に依存するものです．プログラミングにおけるスタイルも同様に，これが正しいとかこれが絶対必要であるというような議論をすることは，むずかしいものです．本文では，C言語でプログラミングをしている多くの先人たちが採用しているスタイルの一部分を紹介しますが，多分に筆者の個性が反映されていることをお許しいただきたいと思います．あくまでもスタイルを決める上での参考になればというのが，本文の目的であるからです．したがって，読者のおかれている環境や目的からすると，必ずしも的を得ていないものもあるかもしれません．読者はこれらを批評して適当に取捨選択し，あるいは発展させて，自分自身の環境や目的に合ったプログラミングスタイルを確立することが大事です．終始一貫したプログラミングスタイルで書かれたプログラムは，自分にとっても読みやすいですし，他人も好感をもって受け入れるものでもあります．

Cのスタイルとは

C言語の特徴からくるスタイル

プログラミング言語には，大ざっぱに分けて3つのスタイルがあります．言語のもつ文法とコンパイラの機能からくるスタイル，言語が実際に使われ発展していく過程でできあがる慣習的なスタイル，そしてプログラムを読みやすくしてドキュメント性を高めるためのスタイルの3つです．はじめの2つのスタイルは言語による依存度が高く，最後の1つはだいたいどの言語についても共通するスタイルです．

それでは最初に，C言語の文法やコンパイラがもっている特徴からくるスタイルについて眺めてみることにしましょう．

C言語の特徴

スタイルから見たC言語のおもな特徴をあげると次のようになります．

プログラムの単位は関数である

C言語のプログラムの単位は関数です．これは他の言語の手続き，サブルーチン，関数副プログラムに対応するものです．関数は入れ子にしてブロック構造にはできません．しかし，{}で括った複文の中で，データの定義・宣言を行うことで関数内でもデータについてはブロック化ができます．

C言語の関数はすべて同一のレベルにあり，とくに意図して不可視にしないかぎり，1つの関数はどの関数からも呼び出せます．mainという名前をもつ関数は，最初に実行される関数であるという1点の例外を除いて他の関数と同じです．関数は再帰的に呼び出せるように書くことも，リエントラントに書くことも容易です．

構造化プログラミングに対応した言語である

連接・選択・反復という構造化プログラミングに対応した制御文が備わっています．トークン（キーワード・識別子・演算子・分離記号・定数・文字列）の間には，任意の数の空白や改行を含めることができるので，かなり自由度の高いスタイルをとることができます．

簡潔な書式を旨とする言語である

他の言語でキーワードとなっているものが記号で表現されたり，簡潔な記述のための演算子があったり，式の値が利用できたりすることにより簡潔な記述ができます．たとえば図1のような書き方ができます．

豊富なデータの型と演算子がある

char, short int, int, long int,

MS-Cエラーメッセージ
Fatal error C 1024: cannot open include file'filename' #includeプリプロセッサ疑似命令内で指定されたファイルが見つからなかった．

float, doubleなど，アセンブリ言語で基本的に扱えるデータの型が完備されています．とくに整数には符号つきと符号なしの区別もできます．これに加えて，それらのデータの格納場所を示すポインタも型として存在します．ポインタは，他の言語と異なりかなり自由に操作できます．ポインタのポインタ，あるいは関数へのポインタも作れますので，見かけ上型は無限に存在することになります．

変数は単純変数のほか，配列，構造体，共用体などがあり，それぞれ型を作ります．他言語にあってC言語に直接備わっていない型は，文字列型，模様型，BCDの数値を表現する型くらいです．

なお，文字列型はありませんが，C言語では可変長の文字列をchar型の配列として扱う最小限の機能があります．

C言語には，他の言語と同様，四則，比較，論理，ビットに対する演算子が完備されています．その上，ポインタに関する演算子，代入および代入演算に関する演算子，3項演算子など，多数の見慣れない演算子が存在します．これらの演算子は式を簡潔に書いたり，コンパクトなオブジェクトプログラムを生成したりするのに有用な演算子です．逆に他の言語にあってもC言語にない演算子は，文字列の操作に関する演算子とべき乗を計算する演算子くらいです．

演算子は変数，定数，関数の戻値などと結びついて式を作ります．ここまでは他の言語と同様ですが，C言語では単純な代入を示す＝もまた演算子であり，そのほかの代入演算子も当然ながら演算子です．たとえば，

```
a=1.0  b+=c  d++
```

などは代入を伴いますが式に分類されます．式はすべて値を求めることができ，この性質をうまく使うと簡潔な記述ができます．

図1　C言語の記述は簡潔である

C言語	他の言語

`static int i=50;`
`a[i++]=b[j++];`
`{........}`
`a=b=0;`
	`dcl i fixed bin(15) static init(50);`
	`a(i)=b(j);`
	`i=i+1;`
	`j=j+1;`
	`begin ........ end;`
	`a=0;`
	`b=0;`

大文字と小文字を区別する言語である

C言語は，キーワードと識別子（変数名・関数名などのユーザーがつける名前）について英大文字と英小文字を別の文字として区別します．識別子に使える文字は英大文字，英小文字，_（アンダースコア），それに数字です．ただし，数字を識別子の最初の文字にはできません．標準的なキーワードは英小文字のみで構成されます．われわれ日本人はあまり感じませんが，欧米人には小文字が好まれるようで，ソースコードに含まれる小文字の比重がかなり高くなります．このスタイルは日本でも尊重され，同じように小文字が多用されています．

コンパイラにはプリプロセッサの機能が標準的に備わっている

プリプロセッサというのは，コンパイルを行う前にソースコードを加工する処理です．条件つきコンパイルのための処理，マクロの定義と展開，指定されたファイルを取り込んでソースコードに併合する処理，関数毎のオプションの指定などの処理が，これらの処理に含まれます．アセンブリ言語の疑似命令，マクロ機能に相当するものと思えばよいでしょう．

プリプロセッサのためにプリプロセッサ文が用意されています．よく使われるプリプロセッサ文は，ヘッダファイルを取り込む# include文とマクロを定義する# define文です．このような機能はC言語でのプログラミングを容易にするばかりでなく，ポータビリティを向上させ，プログラムの再利用を容易にしています．

コンパクトな言語仕様である

C言語そのものには，宣言に関する文，数種類のプログラムの流れ制御に関する文，それに式文があるだけです．文字列の操作に関する文，入出力に関する文などは，C言語の仕様に含まれていません．また，内部関数としての特別な関数の規定も存在しません．これらの仕様に含まれていない機能は，C言語の仕様上すべてユーザーが定義する関数にまかされています．このように他の言語に比べるとC言語はコンパクトな仕様をもつ言語であり，簡単なプログラムを書いてみようという初心者には比較的取りつきやすい言語です．

C言語の仕様に含まれない機能については，実際には個々の関数をいちいち作成する必要はありません．C言語の処理系には入出力，文字列の処理，算術関数など汎用性のある関数がライブラリ関数として必ず付属しています．そして，必要であればC言語自身，あるいはアセンブリ言語でライブラリ関数を書くこともできます．ライブラリ関数の多くはUNIX風の伝統的なものであり，それらの使い方には一種独特のものがあります．これらの関数を使用すれば，入出力，文字列の処理，高度な算術演算などが他の言語と同様に行えることになります．

プログラムの構成

C言語プログラムを書こうとするとき，最初に気になるのがソースファイル上のプログラムの構成です．ここでは，ソースファイル上のプログラムの構成について簡単に説明します．

リスト1に簡単なプログラムの例をあげますので，それを参照しながら説明を読んでください．このプログラムは角度を入力し，それをラジアンに変換します．そして入力された数値，ラジアンに変換された数値，ラジアンに変換された数値に対応するsin関数の値を表示するものです．

C言語のソースファイルの先頭は，/＊と＊/で囲まれたコメントを除けば，たいてい#include文ではじまります．この#include文で取り込まれソースファイルの先頭に併合されるファイルのことを，先頭にあるのでヘッダファイルと呼びます．ヘッダファイルにはライブラリ関数のプロトタイプ宣言(型宣言)，ライブラリ関数に関連するマクロ，型識別子の宣言，グローバルな変数の宣言などが含まれています．

ヘッダファイルには，ライブラリ関数のグループ別に処理系で提供されるものと，ユーザーが独自に開発して汎用化した関数のために作成したものの2種類があります．これらの関数を呼び出す場合，それらの関数に必要なヘッダファイルを先頭で取り込む必要があります．逆にこれらの関数を使わない場合，ヘッダファイルを取り込む必要がないので，#include文は記述されません．

次にあるのがソースファイル内でとくに必要となるマクロの定義，関数のプロトタイプ宣言，外部レベル（関数の定義の外側）でのデータに関する宣言です．これらの順序は一定していませんが，最初の関数の定義の前に記述されます．これらのうち必要のないものは，当然，省略されます．

UNIX上で動作するコンパイラの一部，パソコン上で動作する古いバージョンのコンパイラでは，関数のプロトタイプ宣言を受け入れないものがあります．このようなコンパイラに対するソースファイルでは，プロトタイプ宣言から仮引数の情報を除いた形の関数の型宣言が行われます．ここまでに記述された内容は，少なくともソースファイル内の関数全体に対して効果のあるものです．

次に関数の定義が続きます．複数の関数定義がある場合，関数の定義の順序はまったく任意ですが，一般的にはトップダウンの順序で定義されます．main関数がある場合はそれが最初に定義され，main関数から直接呼ばれる関数がそれに続きます．そしてそれらの関数から呼ばれる関数がさらに続くという順序です．

関数の定義の内側のことを内部レベルと呼びます．内部レベルで記述された内容は，それが記述された関数の内部でだけ有効です．なお，関数と関数の定義の間には，まれにプリプロセッサ文が記述されることもあります．

伝統的なスタイル

K&Rから受け継がれたもの

C言語の伝統的なスタイルは，カーニハンとリッチー(K&R)による著書『プログラミング言語C』(1978)の中

list

▶リスト1

```
/* ヘッダファイルの取り込み */
#include <STDIO.H>
#include <MATH.H>

/* マクロの定義 */
#define PI 3.141592653589793

/* inp_dbl関数のプロトタイプ宣言 */
int inp_dbl(void);

/* 外部レベルの変数宣言 */
static double x;

/* main関数の定義 */
int main(void)
{
    double  rad;

    while(inp_dbl()!=EOF) {
        rad=x*PI/180.0;
        printf("%e[deg]=%e[rad], sin(%e)=%e¥n",
            x,rad,rad,sin(rad));
    }
    return 0;
}

/* inp_dbl関数の定義 */
int inp_dbl(void)
{
    int inpsts;

    inpsts=scanf("%e",&x);
    if(inpsts==1) inpsts= EOF;
    else inpsts=EOF;
    return inpsts;
}
```


MS-Cエラーメッセージ
Fatal error C 1027: DGROUP data allocation exceeds 64 K　64 KBを超える変数がデフォルトのデータセグメントに割り当てられた．

に詳細に解説されています．この著書は，C言語プログラマのバイブル的な存在で，現在もこの著書のスタイルの多くが受け継がれています．この節では，とくに伝統と呼ばれるにふさわしい項目に限って紹介します．

大文字と小文字の使い分け

C言語は，大文字と小文字が区別されることから，ある種の識別子の英文字の部分は大文字だけを組み合わせたものを，ある種の識別子の英文字の部分は小文字だけを組み合わせたものを使う習慣があります．習慣上，英文字の部分に大文字だけが使われる識別子には，

- 記号定数（単純に数値に置き換えられるマクロの識別子）
- typedefで型指定子として宣言される識別子
- 構造体または共用体のタグとしての識別子

があります．

上記の識別子のうち，記号定数はほぼ例外なしに英文字の部分は大文字を組み合わせた識別子が使われます．他の2つは，英文字の部分に小文字を組み合わせた識別子も使用されます．

逆に，英文字の部分に小文字を組み合わせで作られる識別子は，上記以外の識別子で，プログラムの中でひんぱんに現れるものをあげると，

- 関数の識別子
- マクロの識別子(引数を伴うもの)
- 単純変数と配列変数の識別子
- 構造体変数と共用体変数の識別子

などがあります．

英文字の部分に，大文字と小文字の両方を組み合わせて識別子を作ることも，もちろん可能です．しかし，C言語ではこの方法はポピュラーではありません．このことの一因として，リンカの問題があります．グローバルな変数名と関数名はそのままリンカに渡され，アドレスが決定されます．しかし，リンカは必ずしも大文字と小文字を区別するとは限らないので，Cのコンパイラでは区別され何の問題もない識別子が，リンクの段階で混乱をひき起こすことがあります．移植性を重視してきたC言語のプログラマたちは，不要の混乱を避けるために，1つの識別子の中に大文字と小文字が混在することを嫌ったのではないでしょうか．

伝統的な名前をもつ識別子

前述のK&Rの著書にでてくる識別子の多くは，現在でもそのままの名前で使われ，定着しています．そのおもな例を表1に紹介しておきましょう．

ここにあげたもののほか，各処理系には各処理系の伝統的な名前をもつ識別子があります．読者の使用できる2つ以上の処理系のヘッダファイル，あるいはリファレンスをくらべてみるとおもしろいと思います．たとえば，MS-DOS用の処理系には，MS-DOSに向いた伝統的な識別子を発見できるでしょう．

C特有の簡潔な式

「C言語の特徴」の章で述べたように，C言語は簡潔な記述を旨とする言語です．そこでいろいろなプログラムを読んでいくと，他の言語では考えられないような記述によく出会います．この種の記述は，C言語をはじめたばかりの人にはわかりにくいものですが，慣用句のようなものですから数例紹介しておきます．ただし，初心者はここにあげるような例をやみくもに使うのは避けたほうが無難です．多少簡潔さを欠いたとしても，はじめのうちは自分の理解できる方法で記述するのがよいと思います．

例1

リスト2は，標準入力装置からEOFになるまで，1文字ずつchに文字を読み込み，chに関する処理をするものです．リスト3のように書きなおすとわかりやすいでしょう．

例2

リスト4のdstとsrcは，文字列へのポインタです．srcの文字列をすべてdstへコピーします．リスト5のように書いたのと同じです．

例3

リスト6は，ある関数の内部での自

表1　識別子の例

NULL	決して割り当てられることのないポインタの値
EOF	ファイルの終わりに達したことを示す値
FILE	ファイルストリームを制御するための構造体を表す型指定子
stdin,stdout,...	標準入出力装置などのストリームを制御する構造体へのポインタ
argc,argv	コマンドラインの情報を main 関数が受け取るための仮引数
printf,scanf...	入出力関数（マクロ）
calloc,free	ヒープ割当の関数
strcat,strcpy,...	文字列操作の関数
atoi,itoa,...	文字と数値の間の変換を行う関数
isalpha,isdigit,...	文字の種別を検査する関数（マクロ）
exit,system	プロセスを制御する関数

動変数の宣言と初期化の例です．自動変数の初期化は，通常の式文と同じように行われるため，このような記述が可能です．これはリスト7のように，宣言と初期化を別々に行ってもまったく同じことです．

例4

リスト8は，整数の配列xの値を順次標準出力装置に出力します．配列xの各要素の間にはスペースをとり，5要素ごとに改行します．リスト9のように書くとわかりやすくなります．

以上の例のほか，C言語の仕様上省略してもかまわないキーワード，宣言，型キャストなどは，混乱の発生するおそれがないかぎり通常省略されます．たとえば，

- 自動変数の記憶クラス auto
- 関数の記憶クラス extern
- 外部レベルで宣言された変数を関数内部で参照する場合の宣言
- 値を返す関数の値を使用しない場合の型キャスト (void)

などです．

関数の定義の新旧スタイル

伝統あるいは習慣というものは，周囲の状況が変化するにつれ変化していくものです．C言語も一部のエリートたちの言語から，多くのプログラマが使用する言語へと普及していくにつれ，種々の環境で使用できる数多くの処理系が生まれました．そこで問題になったのが，処理系で微妙にくい違うC言語の仕様です．

このような状況の中でC言語に対する標準化の気運が高まり，1989年12月にはANSIの規格として，C言語の標準化が行われました．しかし，この標準化にもかかわらず，現在でもK&R式のスタイルとANSI式のスタイルの両者が共存しています．K&R式スタイルで書かれた膨大なプログラム資産やコンパイラの改版に要する時間のことを考えれば，これはやむをえない状況といえます．

そこで，伝統的なスタイルの最後の

list

▶リスト2

```
while((ch=getch()) != EOF) {
    /* chに関する処理 */
}
```

▶リスト3

```
while(1) {
    ch=getch();
    if(ch == EOF) break;
    /* chに関する処理 */
}
```

▶リスト4

```
while(*dst++ = *src++)
    ;
```

▶リスト5

```
while(1) {
    *dst = *src;
    if(*dst == 0x00) break;
    dst++;
    src++;
}
```

▶リスト6

```
int flsts=fopen(fname, "r+b");
```

▶リスト7

```
int flsts;
..........
flsts=fopen(fname, "r+b");
```

▶リスト8

```
printf("%5d%c",x[i],(i%5 == 4) ? '¥n' : ' ');
```

▶リスト9

```
printf("%5d",x[i]);
if(i%5 == 4) printf("¥n");
else printf(" ");
```

▶リスト10

```
int days_of_year(year, month, day)
int year, month, day;
{
    static int days_per_month[13]
        = {  0, 31, 28, 31, 30, 31, 30,
                31, 31, 30, 31, 30, 31 };
    int days, ii;
    days=day-1;
    for(ii=1; ii<month; ii++)
        days+=days_per_month[ii];
    if(month>2) {
        if(year%4   == 0) days++;
        if(year%100 == 0) days--;
        if(year%400 == 0) days++;
    }
    return(days);
}
```


MS-Cエラーメッセージ
Fatal error C 1031: compiler limit : function calls nested too deeply　プログラムがネストした関数呼び出しのダイナミックコンパイラの限界を超えた．

項目として，とくにスタイル上，目立つK&R式とANSI規格の関数の定義の違いをあげておきます．K&R式では，関数はたとえばリスト10のように定義されます．

そして同じ定義をANSI式で書くとリスト11のようになります．

2つの定義をくらべてみると，違いは一目瞭然でしょう．関数の定義本体はどちらも同じですが，仮引数の宣言の仕方が違います．

K&R式の関数定義では，関数の識別子days_of_yearに続く()の中に仮引数の識別子のみが書かれ，その次に仮引数の型の宣言がなされています．ところがANSI式では()の中に識別子の型宣言を含めた形で書かれます．

そして，この違いは関数の宣言にも反映されます．K&R式では，関数の宣言はリスト12のように，関数の宣言に仮引数の情報は含まれません．一方，ANSI式では，リスト13のように，関数の宣言にはコンパイラが直接引数の型のチェックを行えるよう引数の情報を含めて書きます（関数のプロトタイプ宣言）．ちなみに，ANSI式の関数宣言は関数定義の第1行目をコピーして，最後に；をつければできあがるようになっています．

処理系によっては，K&R式の関数定義を受け入れ，引数の情報を含んだ関数の宣言ができるものがあります．このような処理系の関数宣言は，リスト14のように引数の型だけを指定し，仮引数の識別子は書きません．

このように大きく分けて2つのスタイルがあるのですが，どのスタイルを選ぶかは読者が使用できる処理系に依存します．もし，ANSI式のスタイルを受け入れる処理系が使用できるなら，筆者個人としてはANSI式のスタイルを採用することをお薦めします．

list

▶リスト11

```
int days_of_year(int year, int month, int day)
{
    static int days_per_month[13]
        = {  0, 31, 28, 31, 30, 31, 30,
                31, 31, 30, 31, 30, 31 };
    int days, ii;
    days=day-1;
    for(ii=1; ii<month; ii++)
        days+=days_per_month[ii];
    if(month>2) {
        if(year%4   == 0) days++;
        if(year%100 == 0) days--;
        if(year%400 == 0) days++;
    }
    return(days);
}
```

▶リスト12

```
int days_of_year();
```

▶リスト13

```
int days_of_year(int year, int month, int day);
```

▶リスト14

```
int days_of_year(int, int, int);
```

ANSI式のスタイルは，型のミスによるトラブルをコンパイラ自身が検知して未然に防ぐことができますし，標準化されたことにより，より多くのプログラマが共通の認識をもてるからです．また今後，多くの処理系がANSI式のスタイルを受け入れるようになることも考えられるからです．

読みやすいソースの書き方

人にやさしいプログラミングスタイル

今まではC言語固有のスタイルとでもいうべきものを述べてきましたが，実際に自分でプログラミングを行うとき，具体的にどのような点に注意したら読みやすいソースができるのでしょうか．以下に十分とはいえませんが筆者の選んだ具体例をあげてみます．

構造化プログラミングとC言語

読みやすいプログラミングの基本的スタイルは，構造化プログラミングの理念に従ったプログラミングを行うことです．構造化プログラミングについては多くの参考書があるので，ここでは簡単にC言語との対応を述べるにとどめます．

構造化プログラミングは，連接，選択，繰り返しという3つのプログラムの流れの制御だけでプログラミングを行うことです．C言語での連接とは，文を書いた順序により，それらの文が順次実行されていくというごく自然な形で実現されます．たとえば，リスト15においては，最初にxに0を代入する式文が実行され，次いでfor文による繰り返しが実行され，そしてif文による選択が実行され……，というよう

な実行の流れです．

選択は，ある条件のときにかぎり，ある文を実行するという流れの形態でC言語では図2の文によって実現できます．

繰り返しは，あるグループの文を条件が成立するまで繰り返し実行する流れの制御で，図3の文で実現します．

さらにC言語には，選択と繰り返しの補助的な制御をする文として，図4の2つの文が用意され，これらプログラムの流れ制御を行いやすくしています．

以上のプログラムの流れで，あらゆるプログラムが実現できることが証明されていますが，C言語には現実の問題に，より能率的に対処するため，構造化プログラミングでは使用されないはずのgoto文も用意されています．goto文の問題についてはあとで述べることにします．

fig.

図2　選択

if文	1肢選択（elseのないとき）または2肢選択（elseのあるとき）
switch文	多肢選択

図3　繰り返し

while文
do～while文
for文

図4　補助的な制御文

break文	switch文と繰り返しの文に有効
continue文	繰り返しの文に有効

字下げのルール

実際に構造化プログラミングを行うと，プログラムの流れの制御が複雑な入れ子の構造となって，制御の範囲がわかりにくくなりがちです．そこで考え出されたのが制御の範囲を字下げによって明確にする技法です．C言語においても字下げは必須の技法ですが，いろいろな字下げのやり方があります．リスト16に2つの例をあげてみましょう．

例1と例2とでどちらがよいかという点になると，なかなか優劣つけ難いものがあります．例1では，for文あるいはif文は，当然，制御の開始を示す文だから｛もこれらの文と同じ行に書くべきだというものです．{}のバランスを見るという点では，例2より劣りますが，プログラムに必要な行数が減り簡潔になります．これに反して，例2は{}のバランスはよくわかるのですが，必要な行数は増加します．

どちらを採用するかは読者の好みでよいと思います．ただ肝心なのは，字下げのスタイルを（たとえば例1と例2を）混ぜて使わないことです．

字下げは，コンパイラに対してはま

list

▶リスト15

```
x=0;
for(i=1; i<=y; i++)
    x+=z[i];
if(y != 0)
    x/=y;
..........
```

▶リスト16

例1

```
for(i=0; i<MAXDT-1; i++) {
    for(j=i; j<MAXDT; j++) {
        if(x[i] > x[j]) {
            tmp=x[i];
            x[i]=x[j];
            x[j]=tmp;
        }
    }
}
```

例2

```
for(i=0; i<MAXDT-1; i++)
{
    for(j=i; j<MAXDT; j++)
    {
        if(x[i] > x[j])
        {
            tmp=x[i];
            x[i]=x[j];
            x[j]=tmp;
        }
    }
}
```

▶リスト17

```
for(i=0; i<MAXDT; i++);
{
    sum1 += x[i];
    sum2 += (x[i] * x[i]);
}
```


MS-Cエラーメッセージ
Fatal error C 1033: cannot open assembly language output file 'filename' 指定された名前が有効でないかファイルがオープンできない．

ったく意味をもちません．字下げをしたからといって安心して，リスト17のように書こうものなら，まったく予期しない結果を生じます．

if～else if～else はどう書く

if文はelse ifを続けることによって，選択肢が比較的少ない場合に，switch文と同じく多肢選択に利用されます．この場合の字下げのスタイルは，リスト16の例1に従えば，図5のようになります．例2のスタイルの場合は，図6のようになります．なお，例1の変形として図7のように記述する人もいます．

例1のスタイルを採用した人は図5のスタイルか図7のスタイルを，例2を採用した人は図6のスタイルを使用することはいうまでもありません．

名前のつけ方

プログラムの中で重要な識別子には，ある程度その処理内容や用途がわかるような名前をつけます．たとえば，図8のようなものです．

これに反して，繰り返しの制御変数，一時的なワークエリアなどは，凝った名前を考える必要はなく，自分なりの決まりきった名前をあらかじめ決めておくとよいでしょう．たとえば，図9のようなものです．このようにすることによって重要な名前がひき立ち，プログラムが読みやすくなるものです．

ところで，名前はどんなものでも自由につけられるかというとそうではありません．以下に名前をつけるときに注意しなければいけない点をあげておきます．

①コンパイラによって有効な文字数に制限がある．名前の長さに注意する．

②さらにリンカが識別できる文字数にも制限がある．グローバルな変数名と関数名に注意する．

③リンカによっては大文字と小文字を区別しない．グローバルな変数名と関数名について大文字と小文字の混在を避ける．

④_ではじまる名前はライブラリ関数の内部で使われているものがあるので使用しないほうがよい．

⑤_ではじまり，次に_か大文字の続く名前はANSI規格で予約識別子に指定されている．

⑥ヘッダファイルで定義されている名前と重複する名前は無用の混乱を避けるため使わない．

⑦伝統的な名前は伝統的な用法にしたがって使用する．これらの名前を自分なりに作りかえたり，再定義することは避ける．

fig.

▶図5　字下げのスタイル(1)

```
if(条件式1) {
        ..........
}
else if(条件式2) {
        ..........
}
..........
else {
        ..........
}
```

▶図6　字下げのスタイル(2)

```
if(条件式1)
{
        ..........
}
else if(条件式2)
{
        ..........
}
..........
else
{
        ..........
}
```

▶図7　字下げのスタイル(3)

```
if(条件式1) {
        ..........
} else if(条件式2) {
        ..........
..........
} else {
        ..........
}
```

▶図8　識別子の名前

```
TXT_RED
MAX_DATA_NUM
errmsg_exit
bunsan
file_copy
...
```

▶図9　自分なりの決まりきった名前

```
i
j
k
l
cnt
temp
work
ptr
str
buf
...
```

コメント

各関数の先頭に最低限のコメントをつけておくと，プログラムを読みなおしたり，ドキュメントの作成，その関数の再利用の際に役に立ちます．

例をあげると，リスト18のように書きます．このコメントに含めるものは，関数の名前，引数についての情報，戻値についての情報，機能の概要，注意事項，必要なヘッダファイルなどです．そのほか作成者の氏名，修正履歴などを入れることもあります．読者それぞれの書式で，あらかじめコメントのテンプレートを用意しておくとよいと思います．

関数の先頭以外にも，自分でコーディングしていて苦労したとか，他人にはわかりにくいと思われるところにもコメントを入れておきます．

また，これをして，次にあれをしてというように処理には区切りがあります．そのような箇所に，処理の内容をコメントしておくと，あとで読み返すとき楽になります．たとえば，リスト19のような具合です．

このほか，変数の用途についてもコメントを入れておきます．ただ，各行ごとに必ずコメントを入れたり，処理の区切りを無視したコメントを入れると，かえってプログラムが読み難くなりますので注意しましょう．

list

▶リスト18

```
/******************************************************************/
/*                                                                */
/*  sort_int(data,numb)                                           */
/*  int data[]   :ソートするデータ                                */
/*  int numb     :dataの要素数(data[numb])                        */
/*  [機能]       :dataの各要素を昇り順にソートする                */
/*  [注意事項]   :特になし                                        */
/*                                                                */
/******************************************************************/
void sort_int(int data[], int numb)
{
..........
}
```

▶リスト19

```
/* 年を入力 */
printf("Year = ");
scanf(%d,&year);
/* 月を入力 */
printf("Month = ");
scanf(%d,&month);
/* 月の最後の日付を求める */
if(month==2)    /* 閏年を考慮する */
    last=dpm[2]+adjfeb(year);
else
    last=dpm[month];
/* 月の最初の日の曜日を計算する */
xday=totalday(year, month, 1);
..........
```

▶リスト20

```
*a++=*b++;
```

▶リスト21

```
*a++ = *b++;
```

▶リスト22

```
* a ++ = * b ++;
```

▶リスト23

```
/* 年を入力 */
printf("Year = ");
scanf(%d,&year);

/* 月を入力 */
printf("Month = ");
scanf(%d,&month);

..........
```

▶リスト24

```
ptr1=calloc(NUM1,sizeof(int));
if(ptr1 == NULL) {
    /* 例外処理 */
}
..........
ptr2=calloc(NUM2,sizeof(int));
if(ptr2 == NULL) {
    /* 例外処理 */
}
..........
```

空白の効用

C言語では，トークンとトークンの間には任意の個数のTAB，space，改行があってもよいことになっています．この性質を利用して，プログラムを読みやすくすることができます．

たとえば，リスト20はCのコンパイラに正しく認識されますが，リスト21のように記述すると，演算子を認識しやすくなります．これに，さらに空白を加えてリスト22のようにしてもよいのですが，前の例より多少間延びした感じになります．

いずれにせよ，Cでは演算子に使われる記号が連続することが多いですか

MS-Cエラーメッセージ
Fatal error C 1036: cannot open source listing file'filename'　指定された名前が有効でないかファイルがオープンできない．

ら，適当なグループごと，あるいは演算子ごとにスペースを入れると読みやすくなることは確かです．

改行についても同じことがいえます．リスト19の例でリスト23のように処理の切れ目ごとに改行を入れると，これをして，あれをしてというプログラムの流れが，より一層明確になります．

goto文，使うか使わないか

実際問題として，構造化プログラミングではあまり効率のよいプログラミングができないケースが2つあります．その第1のケースは，関数の中に例外が発生する箇所が複数あり，それらに対して類似する例外処理をする場合です．

たとえば，リスト24のケースでは，同じ例外処理を何回も繰り返して書く必要があります．この同じ例外処理をひとまとめにできれば，作業能率も上がりますし，オブジェクトコードも少なくてすみます．そこでリスト25のようにgoto文が使用されます．

第2のケースは，多重のループから一気に抜け出したい場合です．これも例をあげると，リスト26のような場合です．

ループが何重にもなると，何回もflgのチェックをしなければなりません．このような場合もgoto文を使うと，リスト27のように簡潔になります．

やむを得ずgoto文を使うのは，これら2つのケースにかぎります．goto文は他の制御文と異なり目的がはっきりしませんから，必ず目的がわかるようなラベルを使うか，コメントを入れるようにします．

また，やむをえない場合でも，例にあげたようにプログラムのあとのほうへ流れを移すようにします．この理由は，プログラムを順に読んでいるときラベルが突然現れると，本来のプログラムの流れとは別にもう1つのプログラムの流れが気になり，読む人の集中力が分散するおそれがあるからです．

constを使おう

ANSI規格に準拠した処理系が使用できる読者には，const修飾子が記述できるはずです．constは，宣言に使用して，そのすぐ右側にあるオブジェクトが変更不可であることをコンパイラに知らせます．

たとえば，リスト28は，int型変数aの初期値が128でその値を参照することはできるが，代入演算などで値を変更できないことを宣言しています．もし変数aの値を変更しようとするようなソースコードに出会うと，コンパイラはエラーメッセージであやまりを指摘してくれます．

constを関数の定義とプロトタイプ宣言に使用すると，プログラムの読みやすさが改善されます．たとえば，ライブラリ関数のstrcpyはヘッダファイルstring.hのなかで，リスト29のように宣言されています．この関数は文字列srcを文字列destにコピーする関数ですが，const修飾子により文字列srcの内容を変更しないことを明確に示しています．constは読みやすさの改善とバグの防止の一石二鳥をかね備えた便利な修飾子です．

あとあとのため仕様を残そう

仕様といっても，義務で書かされるような大げさな仕様書のことではありません．数多くの関数をせっせと作っていると，よほど記憶力のいい人でもないかぎり，せっかく作った関数のことをどんどん忘れていくはずです．筆者は年のせいか，あるいは記憶力がもともと不足しているのか，最近このことが深刻な問題になり，悩んだあげく，エディタのマクロ機能を利用して，関

list

▶リスト25

```
ptr1=calloc(NUM1, sizeof(int));
if(ptr1 == NULL)
    goto err_proc;
..........
ptr2=calloc(NUM2,sizeof(int));
if(ptr2 == NULL)
    goto err_proc;
..........
err_proc:
/* 例外処理 */
..........
```

▶リスト26

```
flg=0;
for(i=0; i<NUM1; i++) {
        for(j=0; j<NUM1; j++) {
                ..........
                if(dt == 0) {
                        flg=1;
                        break;
                }
                ..........
        }
        if(flg)
                break;
        ..........
}
```

数の要点がプログラムの編集中にでもワンタッチで簡単に覗けるようにしました。この機能を実現するためには，関数の仕様の要点をファイルに入力しなければならず，簡単な仕様といっても数が多いとたいへんです．筆者は幸いにも，関数の頭に一定形式のコメントをつけていたのでそれを抜き取って加工し，何とかヘルプファイルが完成しました．

これはまったく恥ずかしい話ですが，長い目でみると重要なことですので，敢えて紹介しました．せっかく読みやすいソースコードで汎用性の高いプログラムを書いても，それが眠ってしまうようではもったいないかぎりです．読者の皆さんで，まだこのようなデータベースの作成を実行していない方は，あとあとのためになりますので，ぜひ実行するよう心がけてください．

list

▶リスト27

```
for(i=0; i<NUM1; i++) {
    for(j=0; j<NUM1; j++) {
        ..........
        if(dt == 0)
            goto loop_end;
        ..........
    }
    ..........
}
loop_end:
```

▶リスト28

```
const int a=128;
```

▶リスト29

```
char *strcpy(char *dest,const char *src);
```

むすび
よい悪いの理由を考える

プログラミングスタイルを決めるということは，結局，人間にとって読みやすいプログラムを書くためのセンスを磨くということに通じます．そのためには，いろいろなプログラムを読みあさり，よいスタイル，あるいは悪いスタイルと感じるところがあれば，その理由を考えてみることです．

また，同僚，友人，先輩などに自分のスタイルについて批評を仰いだり，それらの人とスタイルについての議論をすることも必要です．そしてよいと納得できるスタイルを取り入れていくことです．

C＋＋ 処理系紹介

●Turbo C＋＋

AT&T C＋＋ Ver.2.0に準拠．C＋＋のネイティブコンパイラでありながらコンパイル時のオプションによってANSI C対応のCコンパイラとしても使用できる．
製品名：Turbo C＋＋ 1.0／価格：29,800円（'91年3月末まで）
製品名：Turbo C＋＋ Professional／価格：49,800円（'91年3月末まで）
対応機種：NEC PC-9800シリーズ，EPSON PC-286シリーズ（ハイレゾモード対応）／問い合わせ先：(株)ボーランドジャパン 03-3486-1400

●IDEA C＋＋

AT&T C＋＋ Ver.1.2準拠のC＋＋インタプリタ．統合環境からソースプログラムの中間言語へのコンパイル，リンク，中間言語インタプリタの実行，デバッグ等が行える．MIWA C＋＋トランスレータ，MS-Cコンパイラに対応．
製品名：IDEA C＋＋／価格：29,800円／対応機種：NEC PC-9800シリーズ（XA，LTを除くノーマルモードのみ），EPSON PC-286シリーズ／問い合わせ先：三光金属(株)シード事業部 0729-65-7225

●MIWA C＋＋

日本語対応のC＋＋トランスレータ．実行形の作成には別途Cコンパイラが必要．
製品名：MIWA C＋＋ Ver.1.2 (Microsoft C Ver4.0, 5.1対応版)／価格：59,800円
製品名：MIWA C＋＋ Ver.1.2 (TURBO C Ver1.5, 2.0 対応版)／価格：59,800円
対応機種：NEC PC-9800シリーズ，EPSON PC-286シリーズ，富士通FM R-60，IBM5550，東芝J-3100シリーズ／問い合わせ先：(株)ミワ システムズ コンサルティング 045-312-5418

●Zortech C＋＋

C＋＋ネイティブコンパイラ．Windows3.0対応の286用とDOS EXTENDERを含む386用の2種類がある．
製品名：Zortech C＋＋/286 V/2.1開発用パッケージ／価格：168,000円
製品名：Zortech C＋＋/386 V/2.1開発用パッケージ／価格：280,000円
対応機種：NEC PC-9800シリーズ，東芝J-3100,3300シリーズ，IBM PS/55,PS/2，各社AXマシン／問い合わせ先：マイクロソフトウェア(株) 03-3818-8221

MS-Cエラーメッセージ
Fatal error C 1039: unrecoverable heap overflow in Pass 3 ポストオプティマイザのコンパイラパスがオーバーフローを起こし処理を継続できない．

C Reader

4

うまくいかないこともある!?

C言語のデバッグテクニック

平賀礼二

どんなプログラミング言語を使ったかにかかわらず，バグが出ればデバッグが必要になる．このデバッグという作業ほどプログラマを悩ませるものはない．とくにC言語の場合，コンパイルもリンクも無事に終わってから姿を現すバグに悩ませられることが多い．プログラムを起動するたびに症状の違うバグが生じる，不特定に生じる再現不可能なバグ……．これらのバグを追って，徹夜を繰り返したプログラマも少なくないだろう．バグの生じる状況はさまざまであり，これに対処する特定の方法はないといわれている．しかし，C言語で生じるおもなバグの種類と性質を知り，さらに順序だててデバッグを進めることで多くのバグに対処することができるようになる．ここでは，そうした問題を整理したみた．また，プログラマの強い味方となるであろうCodeViewやTurboDebuggerなどの高機能なデバッギングツールについても触れている．

デバッグテクニックの必要性

どうして，誰も教えてくれないのか？

この世の中に，バグのないプログラムは存在しないといわれている．また，プログラミング開発に費やされる時間・労力の大部分は，デバッグに当てられているともいわれている．実際，作成したプログラムが一発でコンパイラを通ることなど滅多にないし，また，たとえコンパイラを通ったとしても，意図した通りに動くことはまずない．結局，デバッグというのはプログラム作成に欠かすことのできない一行程なのである．

では，このプログラマにとって必要欠くべからざる技術について誰かがじっくりと教えてくれるかというと，なぜか誰も教えてくれないのだ．ベテランプログラマの多くは，「デバッガなんかに頼るな，ソースを読め！」とかいうし，学校でプログラミングを教えている教師も「まずしっかりと設計をすることです．バグが見つかったなら，ソースを1行1行丹念に確認しましょう」なんてことしか教えてくれない．

たしかに，机上デバッグをすることは重要である．だが，納期目前，赤い目をして必死のバグつぶしをしているときに，自分のちょっとたりない頭だけがたよりの机上デバッグというのはあんまりである．せっかくコンピュータを使ってプログラミングしているのだから，デバッグにも活用すべきだ．

ここでは，私自身があちらこちらの雑誌やら本やら人から聞いたりして身につけてきたデバッグ法について，簡単に紹介してみたい．ページ数の関係から個々の具体的な内容までは触れないが，初心者がデバッグを進めていくうえでの目安程度にはなると思う．

バグの種類

コンパイルエラー ランタイムエラー

コンパイル時に発見されるバグ

コンパイラに引っかかるようなプログラムのミスをバグと呼ぶことに抵抗を覚えるプログラマも多いと思う．実際，コンパイルエラーになる原因の大部分は単なるタイプミスであって，バグなどという大げさなものではない．しかしながら，ここでは一応これらのエラーも含めてバグと呼ぶことにする．

語句的エラー

語句的エラーは，プログラマがプログラミング言語の法則にあっていないシンボルの使い方をしたときに起こる．つまり，存在しない単語を使ったり，スペルをまちがえることである．

シンタックスエラー

構文エラーともいわれる．これは使用しているシンボルは正しいものの，その並べ方がまちがっているときに起こる．シンタックスエラーは，Cではもっとも一般的なコンパイルエラーである．

実行時に発見されるバグ

実行時エラー

実行時エラーは，プログラマが，文法的にはあっているが，なにか不可能なことや非論理的なプログラムを作ってしまったときに発生する．

C言語のデバッグで一番頭を悩ませられるのが，この実行時エラーである．これらのエラーの中には，入力が変わると症状の変わるバグだとか，printfのようなトレース文をプログラム中に挿入したとたんに症状が消えてしまうバグだとか，起動するたびに症状が変わるバグなどがある．

デバッグの基本事項

探し出し，捕まえ，明らかにし，直す！

デバッグ時の心得

いったんデバッグをはじめたら，そのプログラムを書いたのが自分だということをまず忘れてしまうことである．実際，たとえ自分で書いたプログラムであっても，コードはプログラマの意図どおりになっていないことが多い．したがって，はじめてそのプログラムを見たような気持ちで，コードを徹底的に掘り起こし，試験し，理解してしまうことが肝腎である．

デバッグに取りかかったら頭のスイッチを切り替える，これがデバッグをスムーズに進めるコツである．自分の「意図」に拘泥してしまい，悶々とした時間を過ごすプログラマは少なくない．

MS-Cエラーメッセージ
Fatal error C 1041: cannot open compiler intermediate file - no more file コンパイル作業中に使う中間ファイルが作成できなかった．

デバッグプロセス

デバッグは，けっしていきあたりばったりに進めるものではない．図1のような順序で，整然と行うべきである．

エラーを見つけたなら，まず探索を開始し，そのエラーがプログラムのどの部分に起因しているかを探し，確定する．バグの位置を1つの行かまたは式に限定できればそのバグをつぶすのは簡単である．

コンパイルエラーは，コンパイル時にエラーメッセージが出るので容易に発見できるが，実行時エラーはそう簡単には見つからない．そこで，プログラマの最初の仕事は，作成したプログラムが仕様書通りに動作するかどうかを確認することになるわけである．

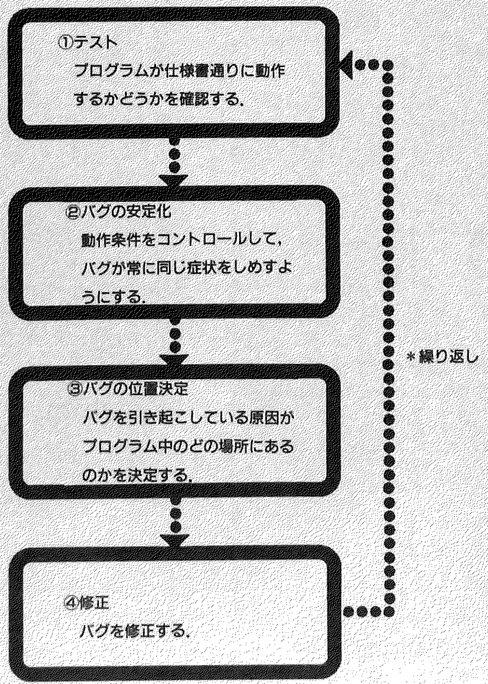
安定化というのは，条件をうまくコントロールして，プログラムにprintf()などを利用したトレース文を付加したあとでも，同じ症状のバグを起すことができるようにすることである．

バグが安定したところで，バグが含まれている可能性のある範囲を狭めていき，特定のセグメント中のコードや変数にバグの位置を決定していく．このときプログラマは，まずデータを収集し，それを解析してなぜそのようなデータができたのかについての仮説をたて，その仮説が正しいかどうかをテストすることになる．

こうして，バグの位置が決定すれば，あとはプログラムを修正してもう一度①のテストを行うのである．

fig.

▶図1　デバッグのプロセス



プログラムのテスト

モジュールごとに，慎重に，記録を取りながら

テストの進行方法

プログラムをテストする方法として，一般にトップダウン法とボトムアップ法の2つがあげられる（図2）．

トップダウン法

トップダウン法は，まず最初にトップレベルのモジュールのテストを行う方法である．このとき，まだテストしていない低レベルのモジュールとして，機能をほとんどもっていないダミーのモジュールを作成して使用する．

ボトムアップ法

ボトムアップ法では，低レベルのモジュールをまず最初にテストする．このとき，それぞれの低レベルモジュールを試験するためにダミーのメイン関数を書く．テストを終了したモジュールを集めて，徐々により完全なサブシステムをつくっていく．

実際のテストでは，トップダウン法とボトムアップ法を組み合わせて使用することが多い．重要なことは，トップダウンかボトムアップかということではなく，テストがモジュールごとにきちんと順々に進められたかどうかということである．

テストの進行は，まずモジュールやシステムの一部のもっとも重要な，正常なふるまいについてテストし，次に特殊なケースやシステムのあまり適用されない部分についてテストするのがよい．

正常なケースがうまくいったら，次に例外的な数値や限界値をテストデータにしてテストを行う．

また，すべてのテスト結果は記録に残し，プログラムの仕様と慎重に比較するべきである．期待値とのどのような相違も潜在的なバグと認め，さらに調査を進める．

限界値

限界値というのは，データ構造の限界付近の特殊な値のことで，この限界値の付近でバグが生じることが多いため，テストデータとして重要な意味をもっている．

モジュールの制御構造をテストしてみることで，限界値をみつけることができる．たとえば，ループ構造では0回とか1回のループだけで終る値が限界値である．また，return, break, gotoで異常停止を起す値も限界値である．算術エラーを起す値も限界値だし，0，－1，1などの数値や大きな正負の数も限界値としてよく使われる．剰余計算では基数に近い数が，ビット演算子を使うときは2のべき乗数が，重要な限界値である．また，リストを操作するモジュールでは，空のリスト，1つの項目しかもっていないリスト，それに大きなリストが限界値になる．

すべてのモジュールが限界値で正しく動作したなら，例外的な数値や起こるはずのないデータでテストを行う．

バグの位置を決定する

プログラムの内部動作を把握しバグを追い詰める

コンパイルエラー

まずソースを読む

コンパイルエラーが出たということは，そのプログラム中に語句的エラーまたはシンタックスエラーが存在しているということである．そこで，プログラマはコンパイラの示したエラーメッセージに従ってソースを見て語句的エラーまたはシンタックスエラーがないかを検討する．当然のことだが，プログラマはそのプログラミング言語のすべての予約語を知っていなくてはならない．

ここで，注意しなければならないのは，コンパイラが告げるエラーの行番号は，いつも誤りが起った行を含む部分の最終行を指しているという点である．ほんとうのエラーは，最初の行からエラー告げられた行までの間のどこかに潜んでいるのだ．

また，エラーメッセージが山のように出たとしても，それらはすべて割り引いて理解しなければならない．正確にバグを指摘しているのは最初の1つか2つのメッセージだけで，あとのメッセージはコンパイラが混乱して発した無意味なものであることが多いからだ．

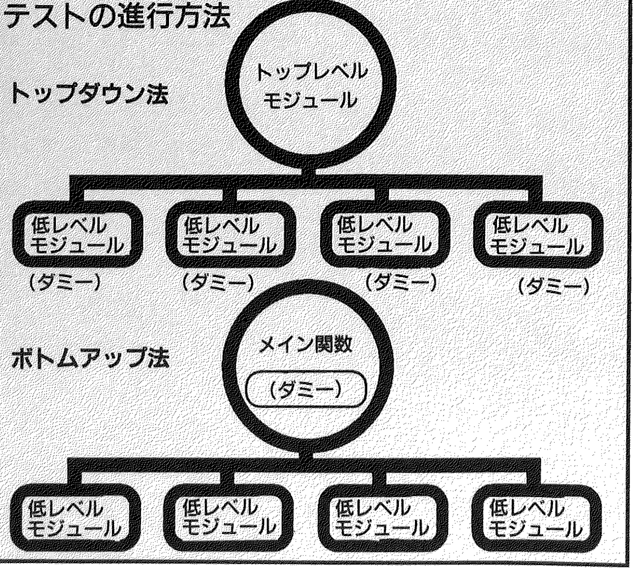
もし怪しいと思われる部分を発見したとしても，あわててコードを変更をしてはいけない．コードを変え，再コンパイルする前に，予想を立て結論を出していなくてはならない．また，コンパイルとコンパイルの間には，たった1つの変更しかしてはならない．こうしないと，いったい自分がなにをしているのかわからなくなってしまう．

ブロックを削除してみる

C言語のプログラムは，文，式，関数定義，関数などからなるブロック構造を形成している．このため，これらのブロックを注意深く削除すれば，プログラムの大きな部分でも小さな部分でも，構文上の整合性を損なわずに取り除くことが可能である．ブロックを削除されたプログラムは動作すること

fig.

▶図2　テストの進行方法



MS-Cエラーメッセージ
Fatal error C 1043: cannot open compiler intermediae file　コンパイル作業中に使う中間ファイルが作成できなかった．

はないが，コンパイルすることは可能である．

完全な構文全体に対応するブロックを切り出して再コンパイルし，エラーが残るようなら，切り出したコード部分にエラーはなかったことになる．エラーが出なくなったとき，残った部分にはエラーコードは含まれていないことになる．こうして，エラーの含まれているブロックを突き止めることができる．

構文エディタ＆中括弧チェッカ

語句的エラーやシンタックスエラーを未然に防止するツールとして，構文エディタおよび中括弧チェッカがある．構文エディタというのは，C言語の構文をチェックし，プログラマが誤ったシンボルを入力したときには入力を拒否するエディタのことである．中括弧チェッカは，Cのブロック構造に用いられている {} の対応をチェックしてくれる簡単なツールである．

どちらも便利なツールなのだが，日本ではあまり流行らないらしく見かけることが少ないのは残念だ．

実行時エラー

printf()を用いたトレース法

printf()を用いたデバッグ手法は，プログラマならば誰でも知っている伝統的な方法である．

この手法は，printf()を用いて作ったトレース文をプログラムの要所要所に挿入し，プログラム実行時の変数およびデータの値を出力させ，プログラム内部の動きをモニタしようというものだ．

この手法を用いる際には，プログラムの構成要素が実行される順序を検証する制御フローのトレースと，データの変化を検証するデータフローのトレースをそれぞれ行うことが多い．

制御フローのトレースは，基本的に制御文の両側にトレース文を置くことで，制御構造が働く前の状態と，プログラムが実行されたあとの制御構造の状態をモニタすることで行う．このとき，常に制御構造の動作を決定する変数または完全式の値を出力するようにする．

関数呼び出しの部分をトレースするときは，関数呼び出しの前後をトレース文で囲むようにする．Cでは，関数が複数の出口をもつので，すべての出口をトレースし，関数が実際にどうやって終るのかを正確にモニタする必要がある．また，関数を囲む必要があるのと同様，パラメータとリターン値はリンク段階の出入りの両側でトレースする必要がある．仮引数，実引数，送られたリターン値，受けとられたリターン値を必ずトレースすることが重要である．

データに着目したトレース文は，データが使われる前と値が代入されたあとにおいて，データの変化をトレースすればよい．

このトレース法で注意すべき点は，プログラム中に挿入するトレース文自体にバグが含まれないようにとくに正確に記述しなければいけないという点である．また，プログラムの全変数に対してトレースを行うことは実際的ではないので，特定の部分の重要な変数についてだけ十分にトレース文を組み込むことが重要である．

チェックサムの応用

プログラムのチェックサムを調べることで，コード上のおかしなポインタを発見することができる．チェックサムは，コードブロック中の各バイトをバイトごとに小さな整数値をもっているとみなして，これを合計して作る．

このチェックサムというのは，もともとはコンピュータどうしをつないで通信を行う際のデータエラーの検出に使われていたテクニックである．

まず，チェックサムを計算する関数を作成し，これをプログラムの要所要所に挿入しておく．次にプログラムを実行し，チェックサムの値を随時出力させる．チェックサムの値が前回に計算したものと異なっていたなら，その間のポインタ参照でコードが変えられたことになる．

このテクニックは，あとで述べる初期化されていないポインタによるバグの発見にも有効である．

C特有のバグ

C言語の自由さは，不安定なバグのもとでもある

Cのもっとも大きな欠点のひとつとして，関数呼び出し時に引数の型チェックを行わないということがいわれてきた．この欠点は，ANSI Cで関数のプロトタイプ宣言が取り入れられたことである程度解消された．これ以外にも，ANSI CではCの言語仕様に多くの改良が加えられた．現在発売されているANSI C準拠をうたったコンパイラは，それ以前のコンパイラに比べ数倍細かいエラーチェックを行うようになっている．

しかし，だからといってC言語がPascalのような厳格なプログラミング言語になったわけではない．あいかわらずCで書いたプログラムは，いとも簡単に暴走してくれる．配列参照をポインタ参照にしてしまう点などは，Cプログラマに大いなる自由を与える

MS-Cエラーメッセージ
Fatal error C 1044: out of disk space for compiler intermediate file　ディスクスペースに余裕がないため，中間ファイルが作成できなかった．

のと同時に難解きわまりないバグをも提供している．これらのバグに対しては，PascalやBASICなどの他のプログラミング言語で使われるデバッグテクニックは通用しない．

さまよえるポインタ

Cは添字が範囲外であるかどうかをテストしない．さらにCでは，範囲外の添字と誤ったポインタとはまったく同じことである．このため，配列の添字が誤ったポインタとしてメモリのなかをさまよいだすことがある．

このような誤ったポインタが，スタックに書き込まれたとすると関数の呼び出しやローカル変数が破壊され，標準ライブラリに書き込まれると演算子や関数の意味が実行中に変わってしまうことになる．最悪の場合にはクラッシュする．

初期化されていないポインタ

Cは，PascalやBASICと違って，ローカル変数が作られた時点で初期値を設定しない．初期化されていないポインタは範囲外の配列よりもたちが悪く，メモリ上のあらゆるところを侵す可能性がある．

初期化されていないポインタのバグは，Cのバグの中でももっとも不安定なバグを作り出す．その特徴は，実行のたびに症状が変わるという点である．

このようなバグのふるまいを安定化するには，グローバル変数，ローカル変数，未割当てのメモリの初期化を行うことが必要である．まずグローバル変数は，C言語の仕様によって0に初期化されるようになっているので特別な処置をする必要はない．次に，ローカル変数の初期化は，そのローカル変数のポインタを利用して行うことができる．また，未割当てのメモリの初期化は，debugやSymdebなどのデバッグ用ツールを使用したり，それ専用の関数を作成してプログラム中に組み込んで行えばよい．

lintの利用

UNIX上でC言語を利用しているプログラマは，lintと呼ばれる構文解析ツールを使ってCコンパイラの欠点を補っている．lintは，プログラム上の技術的にまちがった部分を指摘するのと同時に，技術的には正しいがあまり好ましくない部分についても警告を出してくれる．また，使用されない変数や読まれないコードについても警告情報を出す．

さらに優れたlintは，初期化されていないほとんどの変数のメモリ上の配置をしてくれるので，デバッグプロセスで繰り返しlintを用いることにより，初期化のテクニックは必要がなくなる．

lintは，高度に移植性のあるコードを作り上げるのに有用であるが，ANSI Cによってより精細な言語仕様が利用できるようになったので，現在のコンパイラは，すでにlintで使われていたテスト法の多くを組み込んでいる．

デバッガを利用したマシン語レベルのトレース

Cプログラマにマシン語の知識は必須？

トレース文の挿入や，チェックサムの計算，ローカル変数および未割当てメモリの初期化を行ってもバグを特定できない場合には，デバッガを用いてマシン語レベルのトレースを行う．ここでは，MS-DOSに標準でついてくるSymdebなどのデバッガについて述べ，Turbo DebuggerやCodeViewなどの高機能フルスクリーンソースレベルデバッガについては後述する．

まず最初に，デバッグをはじめる準備として，ロードマップとソースコードをプリントアウトする．次に，ロードマップにリストされている関数のアドレスを逆アセンブルし，コードのトレースを行う．

デバッガにはブレイクポイントと呼ばれる機能があり，この機能を用いてプログラムの実行を中断することができる．あらかじめセットしたブレイクポイントにつながる命令の箇所にくると，プログラムの実行は停止し，デバッガに制御が戻る．

ブレイクポイントで，プログラマは変数やスタックを調べたり，次の箇所へブレイクポイントを移動することができる．プログラムは，いつでもブレイクポイントの箇所から再実行可能である．

ブレイクポイントはマシンコードのどこにでも設けられるが，関数呼び出しやリターンの箇所におくのが普通である．

スタック領域はローカル変数に対して動的に割り当てられるため，ローカル変数の位置を決めるのにロードマップは使用できないが，かわりにローカル変数が宣言されている関数の開始点にブレイクポイントを設けてプログラムを実行し，ブレイクポイントに達したところでマシンのレジスタを調べればよい．

Symdebはたしかに役にたつ道具であるが，けっして使いやすい道具ではない．Turbo DebuggerやCodeViewなどが利用できる環境にあるのならば，無理して使う必要はまったくないといえる．

MS-Cエラーメッセージ
Fatal error C 1047: limit of 'option' exceeded at 'string'　オプションの指定回数が多すぎる．文字列はエラーを起こしたオプションの引数

フルスクリーンデバッガの実際

高機能と使いやすさを実現した標準的デバッガ

Microsoft CodeView

ウィンドウ環境でソースレベルのデバッグ

CodeViewは，MS-Cをはじめとする Microsoft社のプログラミング言語製品に付属してくるデバッガだ．

CodeViewデバッガの特徴の第1は，「ソースレベルでのデバッグができる」ということだ．ソースプログラムの実行をトレースし，実行の停止場所を正確にコントロールすることができるだけでなく，プログラム中で使われるあらゆる変数の値を見たり，これを自由に変更したりすることができる．つまり，CodeViewさえあれば，アセンブラを知らなくてもデバッグできるということにほかならない(もちろん，知識があるにこしたことはないが)．

第2の特徴として，ウィンドウを利用したビジュアルなインターフェースがあげられる．ソースプログラム表示部分，コマンド入力部分，変数表示部分，レジスタ表示部分などがそれぞれ独立したウィンドウとして表示される．また，キーボードだけでなくマウスを使った入力ができるので，ブレイクポイントの設定やウォッチ変数の入力などもその行や変数を直接クリックすることで行える．

CodeViewの画面

まずはCodeViewの画面を見てみることにしよう(画面1)．CodeViewの画面は，Quickシリーズの統合環境によく似ている．画面の最上段にはメニューバーが表示され，その下に次の3種類のウィンドウが表示される．

① Localウィンドウ

現在の関数，またはブロック中でアクセス可能な変数の値を表示する．

② Sourceウィンドウ

通常は，デバッグ中のプログラムのソースコードが表示されている．ブレイクポイント設定行や，トレース時に実行中の行は色つき，反転表示で示される．

Sourceウィンドウは2つまで開くことができるので，ヘッダファイルやほかのソースファイルを参照することもできる．

③ Commandウィンドウ

デバッグコマンドを入力するウィンドウ．

このほかにも，CodeViewの実行中には選択したメニューや入力したコマンドに応じて，次のようなウィンドウが表示される．

④ Watchウィンドウ

ソースレベルデバッガであるCodeViewにはウォッチという機能がある．これは利用者が指定した変数の値をリアルタイムで監視し，画面に表示するという機能だ．これを使えば，画面やプリンタなどの外部出力に頼らなくても，目的のメモリに意図した通りデータが格納されているかどうかを知ることができる．

⑤ Memoryウィンドウ

メモリのダンプリストを表示する．

⑥ Registerウィンドウ

CPUのレジスタ，フラグの内容を表示する．レジスタ，フラグの値は変更することもできる．

⑦8087ウィンドウ

数値演算コプロセッサまたはソフトウェアエミュレータのレジスタ内容を表示する．

CodeViewを使ってみる

では，実際にCodeViewを使ってみよう．キーボードだけでも操作はできるが，マウスを併用したほうが操作はしやすい．基本的な操作は，プルダウンメニューからマウスを使ってコマンドを選択する（これはキーボードでも可能)，キーボードでコマンドウィンドウに直接入力するという2通りだ．

目的のウィンドウにカーソルを移動するには，マウスでウィンドウをクリックするか[F6]キーを使う．

①コンパイル時のオプション

作成したプログラムをCodeViewを使ってデバッグするためには，コンパイル・リンク時にいくつかのオプションを付ける必要がある．

/Zi	CodeViewに必要なシンボル情報を生成する
/Od	すべての最適化を禁止する

ただし，MS-C 6.0付属の統合環境であるPWB(Programmer's Work Bench）を使ってコンパイル／リンクする場合には，デフォルトでこれらのオプションが適用されるのでとくに意識しなくてもよい．

② CodeViewの起動

コンパイルエラーがなくなり，実行形ができたらCodeViewを起動する．コマンドラインから実行するには次のように入力する．

```
A>CV [プログラム名]
```

拡張子は省略してもよい．PWBから実行するときは，Runメニューの

Debugコマンドを選択すれば，自動的にCodeViewに制御が移る．

CodeViewが起動すると，Sourceウィンドウにプログラムが表示される．

③プログラムの実行

[F5]キーを押すと，CodeViewはプログラムの実行を開始し，エラーが発生するとそこで実行を停止する．実行時にプログラムがコマンドラインから引数を受け取る必要がある場合は，あらかじめRunメニューのSet Runtime Argumentコマンドを使って引数を入力しておけばよい（画面2）．

デバッグ

CodeViewを使ったデバッグ方法はさまざまだが，ここでは誌面の都合もあるので基本的な方法を紹介するにとどめる．

・トレース

プログラムをトレース（追跡）して1ステップずつ実行する方法．外見上まったく出力がなく，どこでエラーが発生しているかわからないときなどに利用することができる．Trace [F8]，またはStep [F10]キーを1回押すたびにCのソースプログラムが1行分ずつ実行される．両者の違いはStepの場合には関数の内部まではトレースしないという点だ．

・ブレイクポイント行の設定

ブレイクポイントを設定すると，その行でプログラムの実行を中断させることができる．エラーが発生する前にプログラムを停止させて，エラーの原因となる行が判明するまで1行ずつトレースすることができる．ブレイクポイントを設定するには，設定したい行にカーソルを移動して，[F9]キーを押すか，またはWatchメニューのSet Breakpointコマンドを使う．ブレイクポイントが設定されるとその行は反転表示される（画面3）．

CodeViewでは「ある条件が真になったときだけ」「指定した回数だけ実行したときだけ」「指定した変数の値が変化したときだけ」といった条件付きのブレイクポイントの設定もできる．

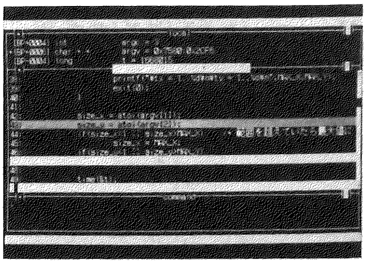
・ウォッチ式の設定

プログラム中の変数をウォッチ式に指定すると，その変数の内容はCodeViewによって監視され，プログラム実行中はウォッチウィンドウにその内容が表示される（画面4）．ウォッチを使えば目的の変数に正しい値が入っているかどうかを確認することができる．ある変数をウォッチ式に指定するには，ソースウィンドウ中でマウスの左ボタンを押しながら変数名をなぞって反転させ，WatchメニューのAdd Watchコマンドを選ぶか，コマンドウィンドウからWコマンドを使って入力する．

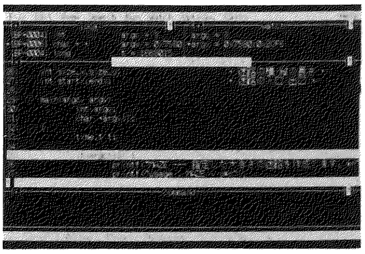
ただし，新しいCodeViewではその時点でアクセス可能なすべての変数のアドレスと値が表示されるLocalウォッチの機能が付け加えられたので，自分でわざわざ変数名をセットしなくてよい場合もある．

エラーの理由がわかったら，FileメニューのExitコマンドでCodeViewを終了してPWBにもどるか，またはエディタを使ってソースプログラムを変更し，もう一度コンパイル，リンク

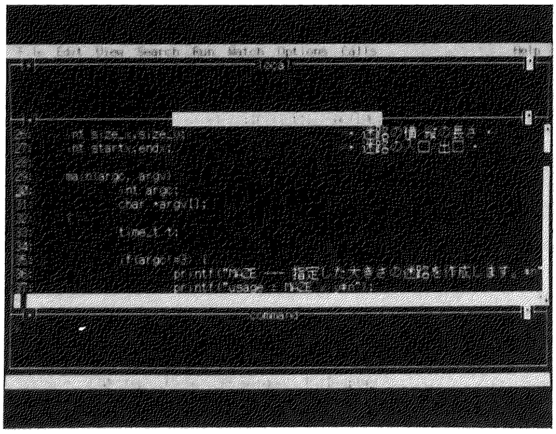
▶画面3　ブレイクポイントが設定された行は反転表示される



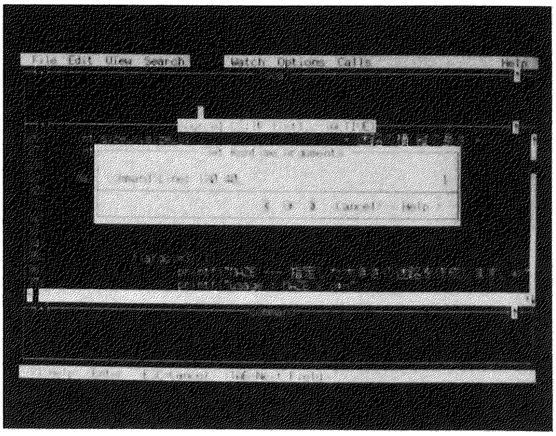
▶画面4　変数の内容がウォッチウィンドウに表示される



▶画面1　起動後のCodeView



▶画面2　Set Runtime Argumentコマンドで引数を入力する



MS-Cエラーメッセージ
Fatal error C 1049: invalid numerical argument 'string'　この文字列ではなく数値の引数が必要．

しなければならない．しかし，CodeViewは前に実行したときに開いたウィンドウや設定したブレイクポイントなどの情報を覚えていて，そのまま再現してくれるので，前回の続きという感覚でデバッグを続けることができる．

ここでは詳しくは触れないが，CodeViewにはほかにも，ウィンドウの内容を印刷したり，デバッグの履歴をとったりといったさまざまな機能がある．また必要に応じて，オンラインヘルプを参照しながら操作できるのも新バージョンの特色である．

Turbo Debugger

巨大プログラムやTSRにも対応

Turbo Debuggerは，Turbo C，Turbo Pascalで知られるBorland International社のソースレベルデバッガである．Turbo Debuggerは，Turbo Debugger&ToolsかTurbo C++，Turbo Pascalの「Professional」という表示のあるパッケージに入っている．

Turbo Debuggerでデバッグできるのは，原則的にBorland社のプログラミング言語製品（Turbo C 2.0，Turbo C++，Turbo Pascal 5.0以上，Turbo Assembler 1.0以上）でコンパイル，またはアセンブルされたプログラムである．付属のユーティリティTDCONVRT.EXE(CodeViewシンボルテーブルコンバータ）を使って，Microsoft社のコンパイラで作成したCやアセンブラのプログラムをコンバートしてデバッグすることもできる．

Turbo DebuggerとCodeViewとは，どちらもソースレベルデバッガであり，CodeViewの2つの特徴はそのままTurbo Debuggerにもあてはまる．

そこで，TurboDebugger独特の部分をみてみると，巨大なプログラムや常駐プログラム(TSR)，デバイスドライバなどのデバッグが考慮されていることがあげられる．

①仮想デバッグ機能

Turbo Debuggerでは80386CPUを搭載したマシンで「仮想デバッグ」と呼ばれる機能が使える．これは，Turbo Debuggerが自分自身を拡張メモリの1MB超の領域にロードして，640KBより下位のメモリ空間をすべてデバッグするプログラムのために解放するものだ．80386CPUには仮想8086モードがあり，80386CPUをいわば複数の8086CPUのように利用できる．Turbo Debuggerでは，TDH386.SYSというデバイスドライバを組み込むことでこれを実現させている．

②保護モードデバッグ機能

仮想デバッグ機能によく似た機能だが，こちらは80286，80386CPUのプロテクトモードを利用する．

③リモートデバッグ機能

2台のパソコンをシリアルポートで結合し，1台でデバッグするプログラムを，もう1台でTurbo Debuggerを実行する機能．Turbo Debuggerのもとではロードできない大きなプログラムのデバッグや，デバイスドライバのデバッグに使用する．

④ TSRのデバッグ

TSRとは（Terminate and Stay Resident)の略で，メモリに常駐して，特定のキーを押すと動き出すプログラムのことで，オンラインソフトとして有名なXSCRIPTやKI-Shellなどはこれにあたる．TSRは実際には常駐する部分と非常駐の部分からなっている．この両方の部分をデバッグするために，Turbo Debbuger自身を常駐させる機能，コンピュータのメモリマップを見るユーティリティなどがある．

Turbo Debuggerの画面

ここでTurbo Debuggerの画面を見てみよう（画面5）．

①メニューバー

画面最上行に表示される．［F10］キーを押すとカーソルがメニューバーに移動する．Turbo Debuggerではこのメニュー（グローバルメニュー）のほかに，ウィンドウごとのローカルメニューがある．

② Moduleウィンドウ

デバッグしているプログラムのソースコードを表示するウィンドウ．ソースレベルでのデバッグを行うときは，このウィンドウ内での操作が主となる．

③ Watchesウィンドウ

ウォッチ式とその内容を表示するウィンドウ．

④クイックリファレンス行

カーソルがウィンドウ中にあるときは，ファンクションキーを使って入力できるコマンドが表示される．［GRAPH］キーや［CTRL］キーを押すと変化する．メニューにカーソルがあるときは，そのメニューについての簡単な説明が表示される．

Turbo DebuggerのViewメニューを使うと，このほかに次のようなウィンドウを開くことができる．

⑤ Breakpointsウィンドウ

設定中のブレイクポイントの表示

⑥ Stackウィンドウ

スタック内容の表示

⑦ Logウィンドウ

メッセージログの表示

⑧ Variablesウィンドウ

アクセス可能なすべての変数の表示

⑨ Fileウィンドウ

ディスクファイルの内容表示

⑩ CPUウィンドウ

レジスタ，フラグ，機械語，16進形式のデータ，スタックの内容を表示

⑪Dumpウィンドウ

指定領域のメモリダンプを表示

⑫Registersウィンドウ

レジスタとフラグのみを表示

⑬Numeric Processorウィンドウ

数値演算コプロセッサに関する情報を表示

⑭Execution historyウィンドウ

実行の履歴を表示

⑮Hierarchyウィンドウ

オブジェクト指向プログラミングで利用

また，変数の値の表示には次のようなウィンドウを開くこともできる．

⑯Inspectorウィンドウ

Dataメニューから Inspectorを選択すると開かれるウィンドウ．選択された変数の現在の値を表示する．

Turbo Debuggerを使う

Turbo Debuggerを使って，Cのソースプログラムをデバッグする場合は次のようになる．

①コンパイル・リンク時のオプション

Turbo Cのコマンドライン版TCC.EXEを使う場合は次のオプションが必要だ．

-v	**Turbo Debugger用のデバッグ情報を生成する．**
-O-	**最適化をオフにする．**

また，TLINKでリンクするときは，/vオプションを指定して，EXEファイルにデバッグ情報を付加する必要がある．

Turbo C++の統合環境版では，Optionメニュー－DebuggerのDebuggerダイアログボックスのSource debuggingボタンをStandaloneに，Turbo C2.0の統合環境版ではDebugメニューのSource debuggingオプションをStandaloneまたはOnにセットする．

② Turbo Debuggerの起動

コマンドラインから起動するには，次のように入力する．

```
A>td [オプション][プログラム名][RET]
```

オプションとプログラム名は省略してもよい．その場合は，Turbo Debuggerが起動してからプログラムを読み込むことになる．ここでは詳しいオプションの説明は省略するが，マウス使用の有無やリモートデバッグなどはオプションで指示する．

デバッグ

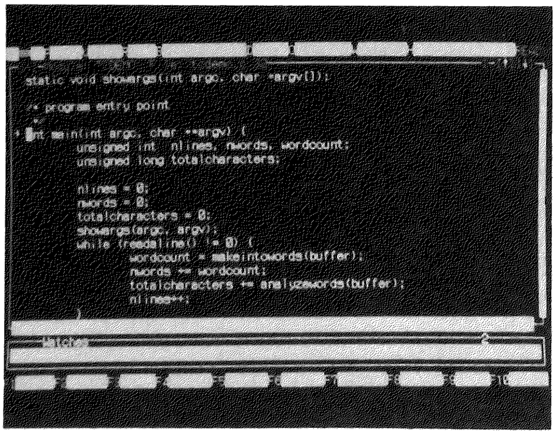
Turbo Debuggerの基本的なデバッグ手順はCodeViewと共通である．つまり，プログラムの実行を制御することと変数などの内容を表示させることである．Turbo Debuggerのインターフェースはファンクションキーを駆使し，できるだけ少ない手順でこれらの操作が実行できるよう工夫されている．

たとえば，プログラム中のデータ構造を調べるInspectなら，目的の変数にカーソルを移動して[CTRL]+[I]キーを押すだけだ（画面6）．

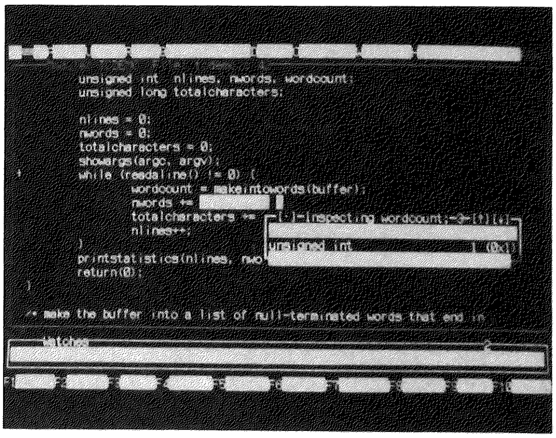
これらの操作のガイダンスは，すべて最下行のクイックリファレンスに表示されていてわかりやすい．ある程度慣れてくれば直感的に操作できるという歌い文句もまんざら嘘ではない．

Turbo Debuggerの機能は，CやPascalで書かれたプログラムのソースレベルでのデバッグにはとどまらず，コンパイラが生成したマシン語のコードを逆アセンブルして調べるといった高度な機能をも備えている．

▶画面5　Turbo Debuggerの起動画面



▶画面6　Inspectウィンドウ



C Reader

5

C言語の明日

オブジェクト指向Cプログラミング

石川雅基・神林靖

新しいパラダイムであるオブジェクト指向が注目されている．その概念はプログラムの設計・開発にとどまらず，データベース，ユーザーインターフェースなどに広く適用されつつある．一方で，UNIXシステムの普及とともに，C言語のユーザーが増加している．これらの状況から，オブジェクト指向のパラダイムをC言語上に実現しようとする試みは自然であり，またどのように融合するかは興味深い問題である．

ここでは，オブジェクト指向の概念を説明するだけではなく，オブジェクト指向プログラムを身近なものとするために，C言語のもつ機能の範囲内でオブジェクト指向風のプログラミングを行い，オブジェクト指向について一層の理解を深めることを目指している．またC言語を拡張することにより，オブジェクト指向プログラムを記述可能とした言語がいくつか発表・販売されている．それぞれの言語によるオブジェクト指向プログラムを取り上げることにより，その特徴およびオブジェクト指向の組み込み方について述べる．対象とする言語はC++，Objective-C，superCである．

C言語について

C言語をオブジェクト指向の観点から見る

C言語は，システム記述が可能な汎用のプログラミング言語である．近年ではそのシステム記述の能力，コンパクトさ，移植性からアセンブラに代わるものとしても利用されている．UNIX上で動作するプログラムの多くがC言語で記述されており，またUNIX自体もC言語で記述されていることで広く知られている．近年のUNIXの普及にともない，多くのアプリケーションがC言語で記述されはじめている．歴史的には，BCPLの影響を大きく受けた言語であり，AT&Tのベル研究所で開発された言語である．

言語の仕様は，「直接アドレスをハンドリングすることができる」「文字列，配列などの比較複写ができない」など，比較的低水準の言語であるが，それらはけっして欠点にはなっていない．むしろプログラムを記述するための自由度を高め，さらに処理系（コンパイラ）をコンパクトにすることに役だっている．入出力の機能をもたないことも，OSへの依存性を下げることに役だっている．

言語の特徴として，

①シンプルな制御機構をもつ（関数のネスト，並列処理などはない）．
②関数の再起呼び出しが可能である．
③ポインタ（アドレス）を陽に扱うことができる．
④bitレベルの操作が可能である．
⑤構造体によってユーザーがデータ構造を定義することができる．
⑥動的なメモリ管理機能をもたず，静的な領域とスタック領域だけを使用する．
⑦値渡しによる引数ひき渡し機能をもつ．

などがあげられる．

①については，Pascalのように関数をネストして定義することができず，またModula-2やAdaのように並列処理の記述をすることもできない．しかしif-else, for, while, do-while, switch-caseなどの制御機構を用いることにより，かつて話題になったgoto文を使用することなく，プログラムを構造化し，記述することができる．②の再起呼び出しは，プログラムを明快に記述する上で強力な機能である．③，④はシステムプログラムのために重要な機能となっている．

C言語をオブジェクト指向の観点から見た場合に，⑤，⑥は興味深い特徴である．また，オブジェクト指向の重要な特徴である情報の隠蔽を考える上で，⑦は重要な性質となる．

たとえば，⑥に関係するstatic宣言ー静的変数を宣言するためのーは，ファイル，あるいはブロック内に情報を隠蔽することができ，データの抽象化と関わりが深い．このことについては，「C言語にオブジェクト指向を用いたプログラム」の章で述べる．

ユーザーがデータの構造を定義できる機能⑤は，データの構造を重視したデータを中心とする考え方からきており，オブジェクト指向の設計に近い立場をとるものである．しかし，C言語ではユーザーが定義したデータ構造に対する演算（手続き）を定義することができず，データ構造の変更に応じて，それらのデータ構造を利用しているプログラムを変更する必要が生じる（図1）．これは，C言語にデータを抽象化する機能がないことに起因し，モジュラリティの高いプログラムの作成を妨げている．

つまり，データ構造の変更によって手続きも変更する必要が生じ，ソフトウェアの再利用，変更に対して強度(robust性)が高いソフトウェアの構築において問題となる．このためにC言語では，再利用可能なソフトウェアは簡単なデータ構造をもつものに限られ，数値計算ライブラリなどの範囲においてのみ，再利用可能なプログラムを作成することができる．

これらは，オブジェクト指向言語がもつデータの抽象化と継承機能などによって，はじめて実現することができる．そのためには，C言語の機能に動的なメモリ管理機能とデータ構造に対する手続きの定義機能の拡張が必要となる．

関数の引数ひき渡し機構として値渡しを採用していることは，モジュラリティを高める上で注目すべき特徴である．値渡しは関数の性質を，手順をまとめたプログラムとしてではなく，固有の機能をもつまとまりに変えることができる．

プログラム全体の流れのなかの一部を別途記述する手続きは，共通の処理をまとめることによりプログラムの開発効率を高めることができる．しかし，このプログラムでは設計者の意識に，手続きの内部と外部との間に強い区別は存在しない．

一方，C言語における関数化は手続きに何らかの機能を対応させることにより，意味をもたせ，プログラムを理解しやすいものとしている．引数の値渡しによる制約は，関数の外部と内部

MS-Cエラーメッセージ
Fatal error C 1054: compiler limit : initializers nested too deeply　イニシャライザのネスティングがコンパイラの限界を超えた．限界値は10から15レベル．

との区別（障壁）を強く意識させることになる．つまり，基本的に引数を通して関数の内部の処理の影響を外部にもち出すことはできない．これによって，プログラマは，プログラムを設計・開発するとき，あるいはプログラムを読むときに，関数の内部と外部を強く意識することから，モジュラリティを高めることができる．

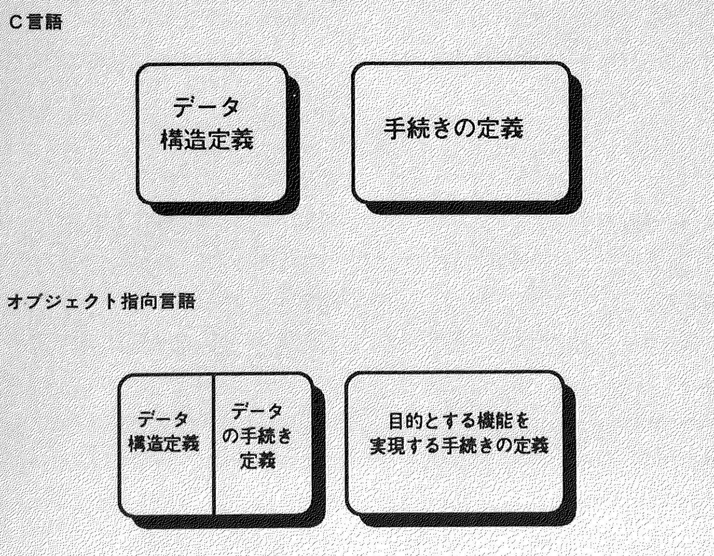
初期のC言語の仕様においては，引数として構造体を渡すことができず，また戻り値として構造体を返すこともできなかった．このため，構造体に関する受け渡しは，アドレス渡しとしてプログラムすることになり，モジュラリティの点では著しい欠陥となっていた（処理系の負荷，実行時の引数のスタック効率を考えると欠点ではない）．しかし，この問題点も最新のANSI規格では改善されている．

このように情報隠蔽の点から，引数の値渡しは重要な機能である．値渡しによる引数の受け渡し機構は，関数の中での影響を外に伝えることができず（副作用がない），プログラムの内容を隠蔽することが可能となる．

ここでわれわれが注目していることは，「いかにモジュラリティを高めるか（隠蔽するか）」「ソフトウェアの再利用をどのように促進するか」「ソフトウェアの変更に強いプログラム構造はどのようなものか」といった点にある．これらを解決する方法として，オブジェクト指向プログラミングが有効であることを次章以降で述べる．

fig.

▶図1　データ構造と手続きの関係



C言語においてデータ構造を変更した場合は，その構造を参照しているすべての手続きに変更がおよぶ可能性がある．しかしオブジェクト指向型の言語ではデータ構造の変更に対して，そのデータ構造に対する手続きの定義を変更することに限られる．

オブジェクト指向

物理イメージを基礎とした抽象化を実現

実用的なソフトウェアを開発する際に，全体をいくつかの扱いやすい単位に分割して作成し，後にそれらを組み合わせてソフトウェアシステムを構成する方式は日常的に使われている．古くはFORTRANのサブルーチンにその起源を求めることができるが，大規模に実施されるようになったのは，ブロック構造をもつプログラミング言語とともに構造化プログラミングが流布してからである．

分割したプログラムの単位は，仕様を明確にすることによって，ほかに再利用したり，不都合があればその部分だけを取り出して交換できるようにすることが望ましい．そのためには，そのプログラムの単位がブラックボックスとなっていることが理想である．

プログラムの単位をブラックボックスにすることを抽象化と呼ぶが，オブジェクト指向プログラミングは，この抽象化の単位として具体的な「もの」，すなわちオブジェクトを採用したものである．先の構造化プログラミングでは，抽象化の単位として手続きが採用された．手続き呼び出しの構文と意味さえわかっていれば，ある手続きを利用するのに，その細部を理解する必要はない．

また，抽象化した手続きにデータ構造を合わせることによって，データ構造も抽象化の単位としたものがデータ抽象である．データ抽象を用いたプログラムも，あくまで能動的な手続きが

受動的なデータ構造に作用するものであり，特定のデータ型に対して手続きを正しく動作させることはプログラマの責任である．

オブジェクト指向の考え方は,「自然界の物理イメージを基礎とした抽象化をプログラミングの世界で実現する」ところにある．オブジェクトは受動的なデータ構造ではなく，それ自身計算パワーをもち，自律的に自身の状態を変化させる計算主体として捉えるのが適当である．

自然界の「もの」を眺めると，ある共通の属性をもった「かたまり」へと分類することができる．この共通の属性を取り出して抽象的な「もの」を作成し，階層化したものがオブジェクト指向プログラミングでいうところの「クラス階層」であり，これにより外的世界の知識を巧みに表現することができる．共通の属性であるデータ構造やそれへの操作（メソッド）を上位のクラスへとくくり出し，下位のクラスでもそれらを使用できるようにして，異なるところだけをプログラミングする．これがクラス継承と差分プログラミングといわれるもので，近年注目を浴びているポリモルフィズムなども素直に実現することができる．

以上の概念を純粋な形でプログラミング言語に実現しているのが，Smalltalkである．そこではすべてがオブジェクトであり，計算はメッセージ送信，つまりメッセージを送ることによって，オブジェクトにオブジェクト内のメソッドの実行を依頼することによって行われる．

C言語をオブジェクト指向化する際には，以上を踏まえて次の機能を含めることが望まれる．

①抽象データ型としてオブジェクトを定義できる．
②オブジェクトによるモジュール化ができる．モジュールには情報隠蔽の機能が含まれていなければならない．
③上位のオブジェクトのデータ構造と手続きを下位のオブジェクトが継承できる．
④ある変数が，文脈によって異なるオブジェクトを参照できるようにポリモルフィズムを支援する．
⑤実行されるメソッドの決定が実行時に行われる動的結合の支援である．

C言語を使用することによって，これらの機能をできるかぎり実現したのがObjective-Cで，C言語をあくまで基本として，それにオブジェクト指向の機能を加えたものがC++であるといえる．

設計方法論としてのオブジェクト指向は，システムをオブジェクトを構成要素として構築することであり，それによって人間の自然な思考をそのままソフトウェアの設計にもち込むことができるようにするものである．次のように特徴をまとめることができよう．

①オブジェクトによるモジュール化によって情報を局所化する．
②オブジェクトを再利用できる部品として扱う．
③オブジェクト間の関係をできるだけ単純にする．

これらの特徴の実現は，オブジェクト指向プログラミング言語を用いれば自然に行えるわけだが，それらに限られるわけではない．たとえば，Cにはアドレスを直接アクセスする機能が備わっているが，これによってポリモルフィズムなどを実現することができよう．これらオブジェクト指向のC言語によるアプリケーションレベルでの実現，そして言語処理系への組み入れについて，次節以降で見ていくことにしよう．

C言語にオブジェクト指向を用いたプログラム

C言語の仕様の範囲でもある程度は可能だが

C言語の言語仕様の範囲で，オブジェクト指向的なプログラムを書くことにより，C言語のもつ機能とオブジェクト指向型のプログラムの記述に必要な機能の関係を明らかにしていく．

ここで例題として,「データを保存するもの」といった抽象的な概念をプログラムとして実現する．このためには,「データを保存する」「データを検索するあるいは取り出す」「データを削除する」といった機能が必要となる．また保存場所をどこにするかといった考えも必要となる．これらを言葉にすると以下のようになる．

保存：
Aどこに　何を　保存する

検索：
どこから　どんなものを　取り出す

C言語でこの概念を実現することを考えた場合に，ファイルのスコープとスタティックな変数を利用してある程度の実現が可能である．

このプログラムを利用するユーザーにとっては，上記にあるような保存，検索といったことにしか興味はない．プログラム作成者は，データをメモリ

MS-Cエラーメッセージ
Fatal error C 1056: compiler limit : out of macro expansion space　マクロを拡張中に内部バッファがオーバーフローした．

上に実装するかファイル上とするかといった，実装上の問題をプログラム内に隠蔽する必要がある．もし通常のC言語の記述法に従えば，データの保存先がファイル上の場合はファイルディスクリプタを，メモリ上の場合はメモリ領域をユーザーが利用する関数の外側で定義する必要がある．

これは，保存という抽象的な概念以外に，その実装に関わることをユーザーが意識することを要求している．このことは，通常のC言語の記述法ではモジュラリティの面で限界があることを示している．

リスト1のプログラムは，static変数を用いることにより実装に関する情報をプログラムの中に閉じ込めている．

リスト1のように記述することより，ユーザーはその実装の仕方に無関係に，プログラムを利用することができる．それはファイルであっても，メモリであっても，ネットワーク経由であっても，さらにはデータベースマネージメントシステムを利用しても同様である．

メモリ上を仮定して実装する場合では，リスト2のようにプログラムすることによって抽象度を保つことができる．

このようなプログラムの特徴は，利用するユーザーが「データを保存するもの」という抽象的な概念を知る必要があるだけで，実際のプログラムの実装の仕方については考える必要がない点である．

ところで，このようなスタイルでC言語を用いてプログラムする場合の限界は，データを保存する場所の確保にある．つまり，このプログラムではデータを保存する場所を複数もつことができず，単一の保存場所を用意することしかできない．これはこのデータ構造を作り出すプログラムが静的であり，実行時に動的に生成できないことによる．

動的な生成を可能とするための方法として，UNIXのもつ動的な生成機能であるforkを利用することが考えられる．しかしfork機能を利用し，プロセスとして実装した場合には，OSによるオーバーヘッドが生じることを考慮しなければならない．また，プロセス間通信が必要であるため，そのためのプログラムを新たに構築する必要が生じる．OSに依存するため，移植性の

list

▶リスト1

```
DataBaseFile.c          /*以下のソースコードを含むファイル*/

static int FileDescriptor;  /*staticとすることにより、隠蔽*/

createDataBase()
{
    FileDescriptor = open("DBFILE",2);
}

int storeData(name,age)
char    *name;
int age;
{
    write(FileDescriptor,・・・
    return(SUCESS あるいは FAIL);
}

int getAge(name)
char    *name;
{
    read(FileDescriptor,・・・
    return(age);
}
```

▶リスト2

```
DataBaseOnMemoryFile.c
static struct{
    char    name[NAMELEN];
    int age;
}buffer[MAX];
int tail;   /* どこまで配列を利用しているかを示す整数 */

createDataBase()
{
    bufferの初期化;
    tail = 0;
}

int storeData(name,age)
char    *name;
int age;
{
    tail++;
    buffer[tail].name
    buffer[tail].age
    return(SUCESS あるいは FAIL);
}

int getAge(name)
char    *name;
{
    for(j=0; j<=tail; j++){
        strcmp(buffer[tail].name,name)==0)
            break;
    }
    検索で目的とするデータがない場合の処理など;

    return(buffer[tail].age);
}
```

低いプログラムとなってしまう可能性もある．これらのことから，あまりよい方法とは考えられない．

そのほかの方法として，関数のアドレス（エントリポイント）を構造体に組み込む方法がある．これは，C言語上で関数のエントリポイントを扱うことができることによっている．

この方法は，リスト3のような構造体を定義することによって行う．リスト3は目的とする概念をプログラムした部分であり，リスト4はそのプログラムを利用している部分である．

リスト4の下線部分がメッセージ送信であり，下線部の記述を先ほどあげた「どれに」「何を」「どうする」に対応つけると以下のようになる．

どこに．どうする（*，何を）

なお，*の部分はC言語で実現するために挿入した引数であり，プログラムの本質ではない．

この場合，さらにデータをファイルで保存する構造体をDBとし，同様の方法でプログラムしたメモリ上で保存する構造体をDB2とすると，プログラマはその内部の実装に関心をもつことなくプログラムを記述することができる．リスト5では2つの異なった構造体を区別することなく利用している．

さらにリスト6では，2つの構造体における関数の格納場所のオフセットを同じとすることにより，それぞれのオブジェクトに対応した関数を実行することができる．先に示した例と同様に，2つの異なった構造体から生成されたオブジェクトdbObjectとdbObject2にデータを保存することを考える．

“保存関数”の中では，渡されたオブジェクトが何であるかを知る必要はなく，オブジェクトに応じた関数が呼び出される．実行時におけるメソッドの束縛（late binding）ではないが，それに類似した機能をこのようにして実現することができる．この機能は，後に述べるC++のvirtual宣言と同等のものと考えることができる．

以上で述べたように，C言語の機能の範囲においても，ある程度のオブジェクト指向プログラミングが可能であることが示された．しかし，これらの方法は言語のレベルでサポートしているものではないため，その記述方法はプログラマに任せられている．このため，記述の方法は大きな自由度を含み，かえって設計のしやすさや，プログラムの読みやすさを損なってしまう．

また，オブジェクト指向型にプログラムをするために，別途ここに述べたようなプログラムを必要とし，プログラマのオブジェクト指向のプログラムを記述する意欲を低下させてしまう．

これらの問題点を解決することを目的として，C言語を拡張した言語が存在している．それらの拡張の特徴について次章で述べる．

list

▶リスト3

```
クラスの定義に当たる部分
struct  DB{
    int fileDescriptor;
    int (*storeData)();
    int (*getAge)();
};

storeData(db,name,age)
struct  DB  *db;
char    *name;
int age;
{
    write(db->fileDescriptor,・・・

}

getAge(db,name)
struct  DB  *db;
char    *name;
{
    read(db->fileDescriptor,・・・

}

DB(db,dbName)
struct  DB  *db;
char    *dbName;
{
    db->fileDescriptor = open(dbName,2,0666);
    db->storeData = storeData;
    db->getAge = getAge;
}
```

▶リスト4

```
クラスの生成およびメッセージ送信に当たる部分
#include        DB.h
main()
{
        struct  DB      dbOject;        /*  インスタンスの生成  */
        DB(&dbObject,"ＤＢ名");          /*  インスタンスの初期化  */

        name = "Ada";
        age = 20;
        dbObject.storeData(&dbObject,name,age);
/*  オブジェクトdbObjectにメッセージstoreDataを送る。この時に保存する内容である
名前と年齢を引数としている。    */


}
```


MS-Cエラーメッセージ
Fatal error C 1059: compiler is out of near heap space　near（デフォルトデータセグメント）ヒープ内に保管するアイテム用ストレージを使い果たした．

list

▶リスト5

```
#include    DB.h
#include    DB2.h

main()
{
    struct  DB  dbOject;    /* インスタンスの生成 */
    DB(&dbObject,"ＤＢ名");     /* インスタンスの初期化 */

    struct  DB2 dbOject2;   /* インスタンスの生成 */
    DB2(&dbObject2,"ＤＢ名");   /* インスタンスの初期化 */

    name = "Ada";
    age = 10;
    dbObject.storeData(&dbObject,name,age);

    name = "Pascal";
    age = 20;
    dbObject2.storeData(&dbObject2,name,age);

}
```

▶リスト6

```
     name = "Ada";
     age = 10;
     保存関数(&dbObject,name,age);

     name = "Pascal";
     age = 20;
     保存関数(&dbObject2,name,age);

 }

 保存関数(obj,name,age)
 struct  DB  *obj;
 char    *name;
 int age;
 {
     obj->storeData(obj,name,age);
```

オブジェクト指向型の言語について

機能を追加したC言語 C++, Objective-C, superC

Smalltalk, Simula, Modula-2, Adaなどのように，言語の仕様自体がオブジェクト指向型あるいは抽象データ型を含んでいる言語もあるが，オリジナルの言語を拡張して，オブジェクト指向型のプログラムを記述可能とした言語もある．たとえば，Lispにオブジェクト指向の概念を加えたflavor, loops, Pascalのobjective-Pascal, PrologのESPなどである．

また，C言語をベースにオブジェクト指向型の機能を追加した言語も数多くある．それらとして，C++, Objective-C, C_talk, superC, COBなどがあげられる．ここではC++, Objective-C, superCについてその特徴を述べる．

C++

C++は，AT&T社のBjarne Stroustrupらによって設計されたC言語を拡張した言語である．よりよいCプログラムを作成することをめざしたものであり，その拡張においてSimulaの影響を受けている．オブジェクト指向を取り込むことを目的として機能を拡張したObjective-C, superCなどと異なり，プログラムの記述力を高めるために機能拡張した結果，オブジェクト指向的なプログラミング要素が付加されたと見ることができる．

Objective-C, superCと同様にC言語の上位互換(super set)であり，C言語で記述されたプログラムは，C++の正しいプログラムでもある．先にも述べたように，C++の拡張目的がC言語のプログラムの記述力を高めることにあり，オブジェクト指向を取り込むことに限っていないため，C++では広範囲にわたって多くの機能拡張がなされている．拡張されたおもな項目をあげると以下の通りである．

①厳密な型チェック機構
②inline展開
③変数の宣言場所
④標準入出力ライブラリ（入出力演算子<<, >>の追加)
⑤クラスの定義
⑥データの動的な生成
⑦導出クラス(derived class, 継承)
⑧メッセージの動的束縛(dynamic binding)
⑨演算の多重定義

そのほかにconst宣言，コメント，無名ユニオン，列挙式(enum)などがある．なお，後半の拡張項目⑤～⑨は，オブジェクト指向型の概念に関連するものであり，オブジェクト指向プログラミングを可能とする拡張である．以下に，それぞれの拡張項目について簡単に述べる．

【厳密な型チェック機構】

従来のC言語では，関数の呼び出し側と関数の定義側の間では，引数と戻

り値に関する型のチェックが厳密には行われていない．C＋＋では，関数の呼び出し形式をヘッダーファイル（＊.h file)に定義し，それをプログラム中に読み込むことにより，定義側と呼び出し側との間の厳密な型チェックを行っている．

【inline展開】

コンパイル時に関数の定義コードを呼び出し部分に展開する機能である．この機能により，呼び出し時のオーバーヘッドを除くことができる．しかし，オブジェクト効率は悪くなる．

【変数の宣言場所】

C言語では，変数の宣言をブロックの最初（手続きの前）に行う必要があるが，C＋＋では任意の場所で変数の宣言を行うことができる．

【標準入出力ライブラリ】

C言語では，入出力に関する機能はライブラリによってサポートされている．この点ではC＋＋も同様であるが，C＋＋では，ストリームライブラリと呼ばれる新たなライブラリが用意されている．このために，提供（定義）されている演算子<<，>>により，入出力の記述を容易にしている．この演算子は多相性をもっており，出力すべきデータのデータ構造にそくした形で入出力が行われる．

オブジェクト指向プログラミングを可能とする上で，⑤～⑨の項目は重要である．オブジェクト指向ではデータを抽象化し，設計者のもつ物理イメージを“もの”(object）としてプログラム中に表現する必要がある．そのためにC＋＋ではデータの抽象化，情報の隠蔽，継承，ポリモルフィズムといった考えを取り入れている．

fig.

▶図2　クラス定義のシンタックス

```
class example :public originExample{        %1
  private:                                   %2
    int value;
  public:
    void print();
    int get();
```

%1)public originExample は class example が class originExample から導出された（継承）ことを示している．
%2)class の宣言内では明記しない限り要素は private と解釈されるので，private の宣言を省略することが可能である．

▶図3　C言語とC＋＋の記述の類似性

```
example Eobj;   //クラスのインスタンス（オブジェクト）の生成

Eobj.print();   //メンバー関数の呼び出し（メッセージ送信）
```

データの抽象化，情報の隠蔽，継承を可能とするために，C＋＋では新たに構造体を拡張した“クラス”を導入している．クラスによって，ユーザーはプログラム中にユーザーの要求を満たす新たな“型”を定義することができる．クラスはオブジェクトの型紙となるものであり，Cの構造体におけるテンプレートと考えることができる．オブジェクトは内部状態をもち，外部からはクラスに用意された処理を通してのみ，その内部状態を変えることができる．

クラスの定義はC言語の構造体の定義に類似しており，シンタックスは図2の通りである．

クラスのメンバは，publicの宣言がないかぎり外部からアクセスすることができない．C＋＋では関数をpublicに宣言することにより，クラスの内部へのアクセスを手続き経由に限定することができる．この関数をメンバ関数と呼んでいる．この関数は，オブジェクト指向でいうメソッドに当たるものである．

クラスの内部の状態にアクセスするために，関数を経由することにより，クラス内部の構造を隠蔽することができ，データ構造の抽象化が可能となる．メンバ関数を実行する形式(syntax)は，「C言語にオブジェクト指向を用いたプログラム」で示したC言語を用いたオブジェクト指向プログラム（リスト4）の形式に類似した記述となる．この記述の類似性は，C＋＋がC言語を拡張することによって得られた言語であることを示すひとつの特徴である（図3）．

C＋＋には，メンバ関数のほかにフレンド関数というものがある．これはクラス用に拡張したC言語の通常の関数と考えることができる．クラス内でフレンド宣言した関数は，クラスのプライベート部にアクセスすることができる関数となり，演算子を多重定義す

MS-Cエラーメッセージ
Fatal error C 1061: compiler limit : blocks nested too deply　プログラム内のブロックのネストがコンパイラの許容範囲を超えた．

る際などに役だつ．

クラスの定義で上位のクラス名を列挙することにより，上位クラスの公開されている属性（メンバ）を取り込むことができる．C＋＋では，このようにして継承を実現しており，継承を利用して作成されたクラスを導出クラス（derived class）と呼んでいる．

継承を利用することにより差分プログラムが可能となり，クラスの再利用を促進することができる（上位クラスのクラス名の前にpublicとあるのは，上位クラスの公開メンバが下位クラスにおいても再び公開されることを示している）．

継承は，ソフトウェアの再利用を考える上で重要なことである．それは完成された部分の信頼性を損なうことなく，プログラマが必要とする機能を自由に追加拡張することができるからである．また，最近のバージョンでは多重継承もサポートされている．

C言語はブロック構造をもつ言語であり，変数にはブロック内で生成される内部変数とプログラムを通して存在する静的な変数の2つがある．もしプログラムの動きに応じて動的に生成される変数を使用する場合は，動的なメモリ管理を行うライブラリ（mallocなど）を利用する必要があった．C＋＋では，新たに演算子new，deleteが用意され，動的なメモリ領域の利用を可能としている．クラスからオブジェクトの生成も，この機能により動的に生成することが可能であり，関数のブロック構造に制約されないオブジェクトを作り出すことができる．

クラスの導入によって提供される機能をまとめると以下のようになる．

①データと手続きをひとかたまりに扱うことができ，データの抽象化が可能となる．
②クラスのメンバをクラス内に隠蔽することができる．情報の隠蔽が可能となる．
③クラスからオブジェクトを生成することができる．
④クラス間の継承により，差分プログラミングが可能となる．

ポリモルフィズムの実現のためにC＋＋では，処理の動的な束縛，データ型に対応した手続きの実行，演算子の多重定義（over loading）などが用意されている．

以下にそれぞれメッセージの動的な束縛，多重定義について述べる．

【動的束縛（dynamic binding）】

動的な束縛は実行時に対象によって，対象にふさわしい処理が行われることを意味する．すなわち，「C言語にオブジェクト指向を用いたプログラム」で述べたように，ユーザーは希望するデータを保存したいのであって，その保存の方法については知る必要がない．これは実行時に対象に応じて，そのオブジェクトに適した処理を決定することによって実現することができる．

C＋＋では，virtualをメンバ関数に宣言することにより，メッセージの動的な束縛がなされる（それ以外の場合は静的に束縛される）．リスト6でとりあげたプログラムに類似した記述により，同一のメッセージに対して異なった処理および変数への操作を行うことができる．

【多重定義（over loading）】

C＋＋では，対象（オブジェクト）だけでなく，実行する関数の引数の数，引数の型によっても実行される処理が異なる．この多重性はプログラムのモジュラリティ，抽象性を高める上で重要な性質である．

さらに，C＋＋では演算も多重定義することができる．これは抽象データ型を説明する上でわかりやすい例題を用意することができる．ユーザーが複素数を定義するとする．その場合に，複素数は実数部と虚数部に分けて定義することができるが，局座標表示を利用することも可能である．つまり，複素数という抽象的な概念に対していくつかの実現法があり，内部的な処理も異なる．しかし，複素数のユーザーにとって内部表現は必要ではない．それにも関わらず，多重性がない場合には，ユーザーは複素数用の＋，実数用の＋（さらには複素数表現1用の＋，複素数表現2用の＋）を使い分けなければならない．これは，ユーザーに扱う対象の内部構造を意識させることにつながり，好ましくない．C＋＋では，演算子を多重定義することにより，同じ＋であってもその型にふさわしい処理が適用される．

他のポリモルフィズムの例として，C＋＋のもつ入出力機能があげられる．C＋＋では，streamへの入出力をリスト7のように書くことができる．

このプログラムは図4の結果を出力する．

このようにユーザーは，変数の型を意識せずにプリントのためのプログラムを簡単に記述することが可能である．これはC＋＋のオペレータがもつポリモルフィズムによって，変数の型に必要な処理を適用し，変数を出力するためである．

このように，C＋＋ではポリモルフィズムにより，データ構造に依存しないプログラムを記述することができ，変更に強く，再利用可能なソフトウェ

アを作成することができる．

ここまでC言語によるオブジェクト指向的なプログラムをベースに，C++の拡張について述べてきたが，リスト8，リスト9，リスト10にC++によるクラスの定義，クラスの継承，メッセージについて載せておく．

リスト8はクラスのインターフェースであり，引数に関する記法は，関数プロトタイピングと呼ばれている（ANSIのC言語規格にも盛り込まれる予定である）．リスト9はメンバ関数の処理の実現法について記述しているものであり，リスト10はオブジェクトを生成してメッセージを送っているところを示している．

▶リスト7

```
float   r=1.1;
int i=10;

cout<<"float r="<<r<<"¥n";
cout<<"int   i="<<i<<;
```

▶図4　リスト7の結果

```
float r=1.1
int   i=10
```

list

▶リスト8　インターフェース部

```
class Point{
  int   aX;
  int   aY;
public:
  Point(int a,int b);
  int getX();
  int getY();
  void move(int a, int b);
};

class Figure{
  Point *origin;
public:
  Figure(int a,int b);
  Point *getOrigin();
  void move(int a, int b);
};

class Rectangle: public Figure{
  Point *corner;
public:
  Rectangle(int x, int y, int a, int b);
  void move(int a, int y);
  int   area();
};
```

▶リスト10　実行部

```
#include    <stream.h>
#include    "rep.h"

main()
{
    Figure aFigure(10,20);
    Rectangle aRectangle(5,10,30,40);

    Point *aPoint;
    aPoint = aFigure.getOrigin();
    cout << "X = " << aPoint->getX() << "¥tY= " <<
            aPoint->getY() << '¥n';

    aPoint = aRectangle.getOrigin();
    cout << "X = " << aPoint->getX() << "¥tY= " <<
            aPoint->getY() << '¥n';
    cout << "area = " << aRectangle.area()<< '¥n';

    aRectangle.move(10,10);
    aPoint = aRectangle.getOrigin();
    cout << "X = " << aPoint->getX() << "¥tY= " <<
            aPoint->getY() << '¥n';
    cout << "area = " << aRectangle.area() << '¥n';
}
```

▶リスト9　実現部

```
#include    <stream.h>
#include    "rep.h"

Point::Point(int a,int b)
{
    aX = a;
    aY = b;
}
int
Point::getX()
{
    return aX;
}
int
Point::getY()
{
    return aY;
}
void
Point::move(int a, int b)
{
    aX += a;
    aY += b;
}


Figure::Figure(int x,int y)
{
    origin = new Point(x,y);
}
Point *
Figure::getOrigin()
{
    return origin;
}
void
Figure::move(int a, int b)
{
    origin->move(a,b);
}

Rectangle::Rectangle(int x, int y, int cornerX, int cornerY)
:(x,y)
{
    corner = new Point(cornerX,cornerY);
}
void
Rectangle::move(int a, int b)
{
    Figure::move(a,b);
    corner->move(a,b);
}
int
Rectangle::area()
{
    return ((corner->getX() - (Figure::getOrigin())->getX())
        * (corner->getY() - (Figure::getOrigin())->getY()));
}
```


MS-Cエラーメッセージ
Fatal error C 1063: compiler limit : compiler stack overflow　プログラムが複雑すぎるためコンパイラスタックのオーバーフローが起こった．

Objective-C

Objective-CをC++から際立たせているものは，その表記法であろう．同じC言語を基本としながらも，構文中にメッセージ送信の機構を取り入れ，オブジェクト指向プログラミングの概念をできるかぎり忠実に実現しようと努力している．

C++がC言語の枠組みの中にオブジェクト指向の機構を追加しているのに対し，Objective-Cは，オブジェクト指向の枠組みをCで実現しようとしているといえる．開発者が言語設計で主たる目標としたのは，ソフトウェア開発の生産性向上であった．そのためのモジュール化の概念的機構がオブジェクト指向であり，それによってモジュールの独立性を高め，再利用が可能なSoftware-ICと呼ばれる部品としている．このSoftware-ICは，次のような特徴をもつ．

①データ構造は1つのモジュール内に密閉されていて，ほかから直接参照されることはない．
②データ構造があるモジュールだけが，そのデータ構造を操作する手続きをもつ．
③モジュールの動作は抽象化されている．
④モジュールはコンパイルされ，再利用可能なバイナリとして提供される．

Objective-Cの環境には標準的なSoftware-ICが付属しているので，Smalltalkで行われているような差分プログラミングも可能である．

Cのような手続き型言語にオブジェクト指向を導入する際，まず問題となるのはクラスをオブジェクトと見なすかどうかである．C++も含めて，多くの型のある言語がクラスをオブジェクトの型と見なしているが，Objective-Cではクラスもオブジェクトとしている．こうすることによって，計算モデルがオブジェクトのみで統一的に構成されるので，理論的に美しくなるだけでなく，クラス変数やクラスメソッドも通常のオブジェクトと同じに扱うことができる．

たとえば，Objective-Cにおけるインスタンスの生成は，クラスにメッセージnewを送ることによって行われるが，これはクラスの定義によってファクトリオブジェクトが作られるので，それにメッセージnewを送るという形式をとっている．したがってクラス定義は，インスタンス変数とインスタンスメソッドのほかに，ファクトリ変数とファクトリメソッド（Smalltalkでいうクラス変数とクラスメソッド）をもつ．ファクトリ変数は，Cの大域変数で実現している．

C++においては，メソッドは通常の関数として実現されているが，Objective-Cでは，メソッドはCにはない概念として別途定義して，オブジェクトへのアクセスはすべてメソッドを介するようにしている．メッセージが送信されたときに起動されるメソッドの決定は実行時に動的に行われるため，メソッドのオーバーローディングができる．ただし，メッセージ式は単項メッセージとキーワードメッセージだけで，二項メッセージはないため，演算子のオーバーローディングはできない．

クラス継承についても，メソッドの継承だけでなく，スーパークラス内のインスタンス変数は，オブジェクトの1部としてアクセスを許すなど，Smalltalkと同じ考え方をしている．

また，C++で実現されている多重継承は，Smalltalk同様許されていない（Smalltalkでも多重継承は可能であるが基本設計は単純継承である）．

モジュールの管理についても，C++はユーザーの責任に任せているが，Objective-CはSmalltalkにあるカテゴリ概念を採用してオブジェクトの整理に当てている．

以上で，Objective-Cの基本的な性格は理解していただけたと思う．基本言語としてC言語を採用しているところは同じでも，Objective-Cは，よりオブジェクト指向の原則に近い言語であるといえる．

リスト11にObjective-Cによる四角形の定義を載せておく．クラス継承の連鎖が，オブジェクト→図形→長方形→正方形となっていて，その表記法がSmalltalkに似ていることに注目していただきたい．

superC

superCは，三菱総合研究所と三菱電機で開発されたオブジェクト指向型のC言語の拡張言語である．Objective-C，C++同様C言語の上位互換となっている．通信系のプログラムを効率よく記述することをめざし，作成された言語である（通信システムのモデルでは階層的に機能を定義し，その機能から作り出される実体をエンティティと呼び，プリミティブと呼ばれるイベントによって，エンティティの内部状態を変更することによって，目的とするモデルを記述する．このオブジェクト指向との類似性から，オブジェクト指向型の言語によって高い記述性が期待できる）．

シンプルであること，十分な処理速

度を得られること，情報の隠蔽が可能なことなどを基本に設計されている．構文の拡張をなるべくシンプルとしたため，superCの構文はC言語との差が非常に小さく抑えられている．

構文上の拡張点は，クラス定義のための構文とメッセージを送るための構文が付加されたにすぎない．このため，C言語のプログラマは大きな障壁もなく，superCでプログラミングすることができる．しかし，オブジェクトの定義，オブジェクトの動的な生成，メッセージの動的束縛，クラスの継承といった機能が提供されるため，オブジェクト指向プログラムを行うのに必要な機能は十分に備えている．

また，superCの特徴として，メッセージの解決を静的に行うように指定できる．これはプログラムが完成した後，そのクラスの処理速度を向上させる際に役だつものである．なお，superCは多重定義をサポートしていない．

以下に簡単にsuperCの構文を説明する．

クラスの宣言は図5の形で行われる．メソッドの定義は通常のC言語の関数表現とほどんど同じであるが，インスタンス変数がスコープの中にある（関数の中からインスタンス変数を参照できる）という点が異なる．

メッセージの構文は図6のように表現する．

メッセージ式は，基本的に動的な束縛によって実行されるが，オブジェクトが何であるか明確な場合はクラスの修飾子を付加することによって，そのメッセージは静的に解決される．この場合はコンパイル時に処理される．

静的メッセージの構文は図7のように表現できる．

用意されている開発のツールとしては，クラスの情報を得るためのブラウザやオブジェクトを対話的に実行できるインタプリタ，クラスのライブラリを含めてコンパイル及びリンクを支援するツールなどをもつ総合的な開発支援環境がある．また，オブジェクトの生成・消滅，オブジェクト間のメッセージの動きを視覚的に表し，対話的にデバッグするビジュアルデバッガも用意されている．

現在，superCはオブジェクト指向としての機能の拡張ではなく，並列処理，分散処理の面からの機能拡張が行われている．この拡張の方向は，オブジェクト指向の並列処理との親和性による．つまり，オブジェクト指向では共有のメモリ領域をオブジェクト間ではもたずに，オブジェクト間のやり取りはすべてメッセージによって行う．

list

▶リスト11

(a)Objective-Cのインターフェース部

```
@interface Point: Object {int x,y;} /* 点のクラス */
+X:(int)a Y:(int)b;           /* ﾌｧｸﾄﾘ･ｵﾌﾞｼﾞｪｸﾄ･ﾒｿｯﾄﾞ */
-moveX:(int)a Y:(int)b;                  /* 点の移動 */
-(int)x;                                 /* x座標 */
-(int)y;                                 /* y座標 */
...
@end

@interface Figure: Object {id origin;}  /* 図形のクラス */
+X:(int)a Y:(int)b;
-origin;                                 /* 基準点 */
-moveX:(int)a Y:(int)b;                  /* 図形の移動 */
...
@end

@interface Rectangle: Figure {id corner;}  /* 長方形のクラス */
+X:(int)a Y:(int)b cornerX:(int)c cornerY:(int)d;
-moveX:(int)a Y:(int)b;
-(double)area;                   /* 面積 */
...
@end

@interface Square: Rectangle {int extent;}  /* 正方形のクラス */
+X:(int)a Y:(int)b extent:(int)e;
-(double)area;
...
@end
```

(b)Objective-Cの実現部

```
@implementation Point: Object        /* 点のクラス */
+X:(int)a Y:(int)b {self = [super new]; x = a; y = b; return self;}
-moveX:(int)a Y:(int)b {x += a; y += b; return self;}
-(int)x {return x;}
-(int)y {return y;}
...
@end

@implementation Figure: Object      /* 図形のクラス */
+X:(int)a Y:(int)b {self = [super new]; origin = [Point X:a; Y:b];
          return self;}
-origin {return origin;}
-moveX:(int)a Y:(int)b {[origin moveX:a Y:b]; return self;}
...
@end
@implementation Rectangle: Object        /* 長方形のクラス */
+X:(int)a Y:(int)b cornerX:(int)c cornerY:(int)d;
    {self = [super X:a Y:b]; corner = [Point X:c; Y:d];
     return self;}
-moveX:(int)a Y:(int)b
    {[super moveX:a Y:b]; [corner moveX:a Y:b]; return self;}
-(double)area {return (double)([self origin x] - [corner x])
                 * ([self origin y] - [corner y]);}
...
@end

@implementation Square: Rectangle        /* 正方形のクラス */
+X:(int)a Y:(int)b extent:(int)e;
    {self = [super X:a Y:b cornerX:(a+e) cornerY:(b+e)];
     extent = e; return self;}
-(double)area {return (double)extent * extent;}
...
@end
```


MS-Cエラーメッセージ
Fatal error C 1068: cannot open file 'filename'　指定されたファイルをオープンできない．

fig.

▶図5　クラスの宣言

```
class  クラス名  上位クラス名 {
        インスタンス変数の宣言
        ・・・
}
{
        メソッドの定義
        ・・・
}
```

▶図6　メッセージの構文

```
オブジェクト名  <  メッセージ:引数  >
```

▶図7　静的メッセージの構文

```
クラス名::オブジェクト名  <  メッセージ:引数  >
```

このことは，オブジェクト単位での並列化が比較的容易に行えることを意味する．現在，ネットワーク（LAN）で接続された複数のワークステーション間でオブジェクトを分散して実行することが可能となっている．

リスト12にC++，Objective-Cで扱った例と同様の図形に関するプログラムを載せる．クラスの定義やメッセージ送信において，先の節であげたプログラムとの類似点がいくつか見られる．一方，異なる点として，superCではクラス構造の定義と実現方法（手続き）を同時に記述する点があげられる．これは，C++のように構造体の拡張としてクラスをとらえず，superCでは手続きと処理のカプセル化ととらえ

list

▶リスト12

```
class Point root{
    int aX, aY;
}
{
    object  new(x, y)
    int x, y;
    {
        return((super<new>)<Point:x, y>);
    }
    object  Point(x, y)
    init    x, y;
    {
        aX=x, aY=y;
        return(self);
    }
    int X()
    {
        return aX;
    }
    int Y()
    {
        return aY;
    }
    move(a, b)
    int a, b;
    {
        aX+=a, aY+=b;
    }
}

class Figure root{
    object  origin;
}
{
    object  new(x, y)
    int x, y;
    {
        return((super<new>)<Figure:x, y>);
    }
    object  Figure(x, y)
    init    x, y;
    {
        origin = Point<new:x, y>;
        return(self);
    }
    move(a, b)
    int a, b;
    {

        origin <move:a, b>;
    }
    object  origin()
    {
        return origin;
    }
}

class Rectangle Figure{
    object  corner;
}
{
    object  new(x, y, a, b)
    int x, y, a, b;
    {
        return((super<new>)<Rectangle:x, y, a, b>);
    }
    object  Rectangle(x, y, a, b)
    init    x, y, a, b;
    {
        super<Figure:x, y>
        corner = Point<new:a, b>;
        return(self);
    }
    move(a, b)
    int a, b;
    {
        super<move:a, b>;
        corner<move:a, b>;
    }
    int area()
    {
        return((corner<x> - (self<orgin>)<X>) *
                       (corner<Y> - (self<orgin>)<Y>));
    }
}
```

ていることの現れである．また，ファクトリメソッド（コンストラクタ）と通常のメソッドの区別がないこともあげられる．

今後の展望

オブジェクト指向型言語の普及はこれから

オブジェクト指向という考え方が，種々のソフトウェアの分野で利用されつつある．オブジェクト指向は，単にプログラム開発の手法にとどまるものではなく，ソフトウェアの設計，ソフトウェアの表現に関係する基本的な考え方として見ることができ，その広がりは多岐にわたると考えられる．ここでは，今後のオブジェクト指向の展望を考える上で，重要と思われるオブジェクト指向要求分析，オブジェクト指向データベース，オブジェクト指向型ユーザーインターフェースについて触れる．

オブジェクト指向要求分析は，上流工程へオブジェクト指向の考えを適用しようとするものであり，構造化分析／設計にかわるものと注目されている．この手法は，オブジェクト指向がもつモデリング能力を利用して，複雑なシステムの要求を表現しようというものである．そのためにデータ抽象，インヘリタンス，カプセル化などの概念を利用している．

オブジェクト指向データベースには，リレーショナルデータベースのように明確な定義はなされていないが，「属性値とそれに対する手続きのカプセル化」「オブジェクトの識別」「永続的なオブジェクト」「データ間のリンクの明示的表現」などの機能が必要とされている．リレーショナルデータベースでは表現できなかったような，複雑なデータの関係やデータタイプを効果的に表現できると期待されている．すでにGemStone, versantといったオブジェクト指向型のデータベースもリリースされている．

オブジェクト指向ユーザーインターフェースは，グラフィックウィンドウ上に存在するものを，それぞれある決まったふるまいをする独立性の高い対象物（オブジェクト）と捉え，現実世界の構造に類似した形でユーザーインターフェースの中に構築しようとするものである．すでにOPEN LOOK, OS/2のプレゼンテーションマネージャなどに取り入れられている．

Smalltalkが提唱したオブジェクト指向という概念が，今，広く受け入れられつつある（Smalltalk自体はそれほど一般化しなかったが）．オブジェクト指向の有効性，C言語およびUNIXの普及を考えあわせると，C言語をベースとしたオブジェクト指向言語によって開発されるソフトウェアは，今後ますます増えるであろう（筆者のまわりでは，ソフトウェアの半分近くがsuperCによって開発されている）．

Smalltalk-80からすでに10年が過ぎている．言語の普及には10年近い時間がかかるといわれているが，オブジェクト指向型言語が一般化するのはそろそろのようである．

参考文献

(1)山本喜一，オブジェクト指向とC言語，Computer Today，No.40，1990.11

(2)神林靖，石川雅基，オブジェクト指向プログラミング，コンピュータアクセス，1989.10

(3)上村努，手続き型言語へのオブジェクト指向の導入，情報処理，Vol.29，No.4，1988.4

(4) Bjarne Stroustrup, C++ Programming Language, Addison-Wesley Publishing Company

(5)オブジェクト指向入門，二木厚吉監訳，アスキー出版局 (Bertrand Meyer, Object-oriented Software Construction)

(6) Brad J.Cox, Object-Oriented Programming An Evolutionary Approach

MS-Cエラーメッセージ
Fatal error C 1070: mismatched #if/#endif pair in file 'filename'　#if,#ifdef,#ifndef疑似命令を見つけたが，対応する#endif疑似命令が見つからない．

C言語関連書籍一覧

わかりやすさを追求した入門書
おかたい標準テキスト

BASIC to C－BASICユーザーのためのC言語入門（海外ブックス）
R.トライスター著，村井純監修／アスキー／A5判／216ページ／1,650円
BASIC/C変換ハンドブック
R.J.トライスター著，矢吹道郎監修／啓学出版／A5判／248ページ／2,060円
BASICからC言語へのステップ（国家試験シリーズ109）
植田一弘著／弘文社／A5判／216ページ／1,550円
BASICからの"C"入門
竹内銀平・野島晋著／技術評論社／A5判／240ページ／1,900円
C++：オブジェクト指向プログラミング（A-W/TOPPAN情報科学シリーズ）
R.ウィナー著，前川守訳／トッパン／A5判／360ページ／3,200円
C++プログラミング（C STEP UPシリーズ2）
門内淳・赤堀一郎著／ソフトバンク／B5判／272ページ／2,680円
C++初級プログラミング
石島辰太郎・小山裕司・大前昭博著／啓学出版／B5判／188ページ／1,650円
C++入門－次世代C言語によるプログラミングの実際
足立高徳著／CQ出版／A5判／196ページ／1,550円
Play the C（上巻）（初級C言語講座第1巻）
林晴比古著／ソフトバンク／A5判／240ページ／1,550円
Play the C（下巻）（初級C言語講座第2巻）
林晴比古著／ソフトバンク／A5判／235ページ／1,550円
Q&Aで学ぶC言語
J.バードン著，川崎盛美訳／啓学出版／B5判／208ページ／2,370円
UNIX流Cプログラミング入門
F.リチャード.ムーア著，中村和郎訳／HBJ出版局／B5判／196ページ／2,369円
C－プログラミング改訂2版
J.パーダム著，石原孝一郎監訳／丸善／A5判／302ページ／2,678円
C－初級入門
堀内征治著／啓学出版／A5判／288ページ／2,700円
Cアンサー・ブック
C.L.トンド＆S.E.ギンペル著／矢吹道郎訳／啓学出版／A5判／204ページ／1,860円
Cアンサー・ブック（第2版）
C.L.トンド＆S.E.ギンペル著，矢吹道郎訳／啓学出版／A5判／246ページ／2,000円
Cコンパニオン
A.I.ホルブ著，村上峰子訳／オーム社／A5判／312ページ／2,987円
Cノート（ソフトウェア工学ライブラリ11）
C.T.ザーン著，ディジタルコンピュータ訳／近代科学社／A5判／120ページ／1,339円
Cハンドブック
M.I.ボルスキー著，石田晴久訳／共立出版／A5変判／84ページ／2,163円
Cビギナーズテキスト
松尾有祐著／翔泳社／A5判／274ページ／1,854円
Cプログラミング（岩波コンピュータサイエンス）
石田晴久著／岩波書店／A5判／232ページ／2,270円
Cプログラミングの基礎
広田豊彦著／培風館／A5判／240ページ／1,854円
Cプログラミングガイド
小野塚純夫著／コロナ社／A5判／208ページ／1,854円
Cプログラミングノート
河西朝雄著／ナツメ社／B5判／263ページ／2,580円
Cプログラミング技法
河西朝雄著／ナツメ社／B5判／278ページ／2,580円
Cプログラミング作法（Information & Computing第12巻）
木下恂著／サイエンス社／A5判／368ページ／2,900円
Cプログラミング入門
河西朝雄著／ナツメ社／B5判／320ページ／2,580円
Cプログラミング入門
石田晴久著／共立出版／A5判／184ページ／2,060円
Cプログラミング入門
中津山幹男著／培風館／A5判／224ページ／1,730円
Cプログラム［入門編］（ソフトウェア・ライブラリ2）
前田英明著／サイエンス社／A5判／272ページ／2,575円
Cプログラム入門
大原茂之著／オーム社／A5判／200ページ／1,648円
C解説書（情報処理基礎講座）
松尾三郎監修／電子開発学園／エスシーシー（発売）／B5判／234ページ／2,060円
C言語
河西朝雄著／ナツメ社／B5変判／268ページ／1,800円
C言語110番
伊藤英治著／ラジオ技術社／A5判／216ページ／1,750円
C言語Q&A
竹田仰著／工学社／A5判／368ページ／2,575円
C言語－Pascalからの入門
D.ブラウン著，西田親生・五月女健治・梅澤正弘訳／啓学出版／A5判／160ページ／1,650円
C言語の活用理解

▶BOOK LIST

保坂光・若林淳著／ソフトバンク／A5判／252ページ／2,060円

C言語の基礎知識
門内淳・乗松保智著／ソフトバンク／A5判／320ページ／2,580円

C言語プログラミング
広内哲夫著／啓学出版／A5判／344ページ／3,200円

C言語プログラミング入門
平野浩太郎・宇佐美悟著／コロナ社／A5判／226ページ／2,650円

C言語プログラミング入門［UNIX版］
西田親生・五月女健治著／啓学出版／A5判／254ページ／2,480円

C言語マスター基礎編
藤本健著／電波新聞社／B5判／208ページ／2,000円

C言語マスター入門編
藤本健著／電波新聞社／B5判／208ページ／2,060円

C言語実用数値処理プログラム
吉川敏則著／近代科学社／A5判／320ページ／2,884円

C言語入門
高澤嘉光監修，北川一明・北川誠二著／共立出版／A5判／240ページ／2,400円

C言語標準教科書
佐々木彬夫・坂巻佳寿美著／共立出版／A5判／260ページ／2,472円

いちばんやさしいC
平林雅英著／日本実業出版社／B6判／224ページ／1,340円

いちばんわかりやすいC言語入門
川崎盛美著／ナツメ社／B5判／224ページ／2,000円

ねことおじさんのらくらくマスターC言語
押野崇芳著／ナツメ社／菊判／246ページ／1,500円

はじめてのCプログラミング
川崎盛美著／ナツメ社／B5判／352ページ／2,580円

はじめてのC言語
佐々木整著／秀和システムトレーディング／B5変判／270ページ／1,300円

はじめての人のための標準C言語プログラミング教本
S.G.コーチャン著，西川利男監訳／HBJ出版局／B5判／320ページ／2,900円

はじめて学ぶC言語
松沢茂著／共立出版／B5判／192ページ／2,575円

ザ・C（NSライブラリ4）
戸川隼人著／サイエンス社／A5判／214ページ／1,957円

パソコンCプログラミング（マイコン活用シリーズ）
T.A.ギブソン著，C研究会訳／東海大学出版会／A5判／200ページ／2,266ページ

パソコンユーザのためのC言語入門
R.トライスター著，鑰山徹訳／共立出版／B5変判／118ページ／1,957円

ビジネスC言語入門
斉藤孝，三和義秀著／日本経済新聞社／A5判／192ページ／2,100円

プライマリーC言語ノート
長谷川孝明著／HBJ出版局／B5判／232ページ／2,400円

プログラミング言語C++
B.ストラウストラップ著，斉藤信男訳／トッパン／A5判／380ページ／4,017円

プログラミング言語C第2版　ANSI規格準拠
B.W.カーニハン＆D.リッチー著，石田晴久訳／共立出版／A5判／360ページ／2,800円

プログラミング言語C－UNIX流プログラム書法と作法
B.W.カーニハン＆D.リッチー著，石田晴久訳／共立出版／A5判／264ページ／2,575円

マスターCハンドブック
C.ボロン著，寅市和男訳／オーム社／B5判／368ページ／5,047円

マニュアルを読む前のC言語入門
涌井良幸・涌井貞美著／誠文堂新光社／A5判／254ページ／1,850円

演習Cプログラミング（セミナーライブラリ計算機4）
吉岡善一著／サイエンス社／A5判／216ページ／1,860円

応用C言語（アスキー・ラーニングシステム）
三田典玄著／アスキー／B5変判／327ページ／2,060円

改訂新版C言語入門（ソフトウェア・サイエンス）
L.ハンコック＆M.クリーガー著，アスキー出版局監訳／アスキー／A5判／282ページ／2,580円

改訂第2版はじめての"C"（ソフトウェアテクノロジー2）
椋田實著／技術評論社／A5判／264ページ／1,900円

基礎からのC言語入門
作田仰・丈和正信著／工学社／A5判／336ページ／2,575円

基礎レッスン"C言語"（下）基礎習得編
椋田實著／技術評論社／A5判／288ページ／1,900円

基礎レッスン"C言語"（上）入門準備編
椋田實著／技術評論社／A5判／296ページ／1,900円

実習C言語（アスキー・ラーニングシステム）
三田典玄著／アスキー／B5変判／288ページ／1,850円

初めての人のC言語入門
JTM企画（株）・荒瀬遙著／西東社／A5判／216ページ／1,300円

初心者のための実習C言語
東藤生著／新紀元社／A5判／308ページ／2,000円

詳説C言語
S.ハービソン＆G.スティール著，斎藤信男監訳／ソフトバンク／A5判／480ページ／4,500円

入門ANSI C
H.シュルト著，久野茂訳／マグロウヒル出版／A5判／392ページ／2,900円

入門C言語（アスキー・ラーニングシステム）
三田典玄著／アスキー／B5変判／224ページ／1,550円

標準C基礎コース（新情報テキスト）
杉山行浩著／東京電機大学出版局／A5判／208ページ／2,060円

標準C発展コース（新情報テキスト）
杉山行浩著／東京電機大学出版局／A5判／192ページ／2,060円
新C言語入門ビギナー編（C言語実用マスターシリーズ1）
林晴彦著／ソフトバンク／B5変判／256ページ／1,900円

目的がはっきりしている応用書 あるいはもっと腕を磨きたい人に

実用C言語ー関数の作り方・使い方
中田利秋・串崎充著／オーム社／A5判／144ページ／1,545円
MIDIのためのCプログラミング
J.コンガー著／工学社／B5判／240ページ／2,500円
TURBO Cによる科学計算入門
平田邦男著／共立出版／B5変判／216ページ／3,100円
Cー言語とプログラミング（コンピューターサイエンス・ライブラリ）
米田信夫著／産業図書／A5判／214ページ／1,957円
Cによる科学技術計算ー複素数の扱いからFFTのプログラミングまで
小池慎一著／CQ出版／A5判／340ページ／2,370円
Cによる数値計算
竹本宣弘・荒実著／朝倉書店／A5判／260ページ／3,296円
Cによる有限要素法のプログラミング
黒木健実・竹迫一雄・添田朋子・大崎眞美子著／森北出版／菊判／216ページ／2,472円
Cの話（PC-PAGE15）
安良城勝也著／翔泳社／A5判／232ページ／1,500円
Cサンプル＆ツール集ートレーニングから実践ツールまで（ソフトウェアテクノロジィ1）
C言語研究会著／技術評論社／A5判／220ページ／1,900円
Cソフトウェア・キット
石島辰太郎著／啓学出版／B5判／248ページ／2,890円
Cツールライブラリースクリーンエディタから3Dグラフィックス（ソフトウェアテクノロジィ3）
C言語研究会著／技術評論社／A5判／258ページ／2,100円
Cパズルブック（ソフトウェアバンク）
A.フューアー著，村井純訳／アスキー／B5変判／184ページ／2,060円
Cプリプロセッサ・パワー
林晴比古著／ソフトバンク／A5判／224ページ／2,270円
Cプログラミング・ガイドライン
T.プラム著，西田親生・五月女健治・長井秀憲訳／啓学出版／A5判／152ページ／1,550円
Cプログラミングの実際ープログラミングの理解と応用
岡元敏雄著／パーソナルメディア／A5判／272ページ／2,266円
CプログラムブックI（ソフトウェアバンク）
打越浩幸・濱野尚人・梅原系著／アスキー／B5判／296ページ／2,880円
CプログラムブックII（ソフトウェアバンク）
打越浩幸・濱野尚人・梅原系著／アスキー／B5判／352ページ／2,880円
CプログラムブックIII LISP処理系の作成（ソフトウェアバンク）
小西弘一・清水剛著／アスキー／B5判／320ページ／2,880円
Cライブラリー
ジャムサ著／戸内順一訳／マグロウヒル出版／B5判／264ページ／2,987円
C言語／ディジタル信号処理
秋月影男・松山泰男著／培風館／A5判／224ページ／1,730円
C言語とISAMーファイル管理入門
植田一廣著／工学図書／A5判／180ページ／1,854円
C言語とMODEM
植田一廣・黒木直樹著／工学図書／A5判／136ページ／1,500円
C言語とRS-232/GP-IB
磯辺俊夫著／工学図書／A5判／200ページ／2,060円
C言語とWINDOW
植田一廣・黒木直樹著／工学図書／A5判／120ページ／1,339円
C言語とアセンブラ
磯部俊夫著／工学図書／A5判／224ページ／2,060円
C言語によるデバイスドライバの作り方（ラジオ技術選書192）
成田強著／ラジオ技術社／A5判／222ページ／2,060円
C言語によるパソコンCAD・CG
C&CG研究会グループ編／工業調査会／B5判／160ページ／1,854円
C言語によるマイコン制御演習
桐山清著／共立出版／B5判／148ページ／2,230円
C言語による科学技術計算サブルーチンライブラリー
黒瀬能聿・松島勇雄・松尾俊彦著／啓学出版／B5判／256ページ／2,990円
C言語による数学解析
落合豊行著／近代科学社／A5判／186ページ／2,200円
C言語のためのデバッグ入門
浅野理森著／ナツメ社／菊判／293ページ／2,200円
C言語のデバッグ手法
R.ウォード著，川崎盛美訳／啓学出版／B5判／208ページ／2,370円
C言語の応用50例ー実践からのアプローチ（C再挑戦シリーズ第3巻）
上原哲郎・石田秋也・乗松保智・中山雅彦・高木康宏著／ソフトバンク／A5判／280ページ／2,369円
C言語の本格的応用
千葉宗昭著／CQ出版／B5判／224ページ／1,545円
C言語パソコンCAD・CGプログラム集
C&CG研究会グループ編／工業調査会／B5変判／260ページ／1,957円
C言語プログラミング500題
田中廣著／日刊工業新聞社／B5判／260ページ／2,270円
C言語プログラミング・ツール

▶BOOK LIST

西田親生・五月女健治著／啓学出版／B5判／300ページ／3,710円

C言語実務プログラミングー構造化モジュール設計によるシステム開発
斉藤孝著／HBJ出版局／B5判／391ページ／3,296円

C言語実用ツールブック
村上晶信著／新紀元社／B5判／363ページ／3,200円

アンダスタンディングC
B.H.ハンター著，小池慎一訳／CQ出版／A5判／308ページ／2,369円

応用C言語
シュルト著，横山直樹訳／マグロウヒル出版／B5判／398ページ／3,605円

困ったときのC言語活用100選
涌井良幸・涌井貞美著／誠文堂新光社／A5判／264ページ／1,850円

上級・C言語の応用50例
上原哲郎・中山雅彦・石田秋也・井上哲理著／ソフトバンク／B5判／296ページ／2,480円

役立つプログラムC言語グラフィック入門
涌井良幸・涌井貞美著／誠文堂新光社／A5判／222ページ／1,850円

処理系に依存した使い方の本　すばやく引けるリファレンス

Lattice C J4.0ハンディ・マニュアル（ハンディ・リファレンス34）
川崎盛美著／ナツメ社／B6判／376ページ／1,450円

Lattice C ver.J3.1/pc-9801 Cツールボックス
W.J.ハント著，石島辰太郎・板倉秀清訳／啓学出版／B5判／384ページ／4,020円

Let's C（Ver3.0）プログラミングハンドブック
PMC研究所編／パーソナルメディア／A5判／382ページ／2,575円

MS-C Ver.5.1オンラインリファレンス
谷口崇著／翔泳社／A5判／176ページ／2,200円

MS-C Ver.5.1マイティリファレンス（ラジオ技術選書202）
菱木研二・斉藤直司著／ラジオ技術社／B5判／520ページ／5,000円

MS-C Ver5.1入門
黒瀬能聿・松島勇雄著／啓学出版／B5判／280ページ／2,700円

MS-C ver.5.1プログラミング
石田秋也著／ソフトバンク／A5判／480ページ／3,300円

MS-Cの使い方
吉川敏則著／工学社／B5判／450ページ／3,090円

MS-Cハンドブック
アスキー書籍編集部編／アスキー／B5判／428ページ／3,610円

MS-Cプログラミングガイド
松坂邦生著／ラジオ技術社／A5判／216ページ／1,750円

MS-Cライブラリ・テクニック
A.ハンセン著，福崎俊博訳／アスキー／B5判／448ページ／3,610円

Microsoft C Ver.5.1ハンディ・マニュアル（ハンディ・リファレンス）
堀越一雄著／ナツメ社／B6判／336ページ／1,300円

Microsoft C初級プログラミング入門(上)（Microsoft Language シリーズ①）
河西朝雄著／技術評論社／A5判／432ページ／2,300円

Microsoft C初級プログラミング入門(下)（Microsoft Language シリーズ②）
河西朝雄著／技術評論社／A5判／320ページ／2,000円

PC-9800シリーズ初級TURBO C
堀越一雄著／電波新聞社／A5判／448ページ／1,854円

QuickC 2.0プログラミング入門
矢部和博著／アスキー／B5変判／296ページ／1,800円

QuickC ver.2.0ランタイムライブラリリファレンス
マイクロソフト監修／翔泳社／A5判／375ページ／2,400円

QuickCハンディ・マニュアル
押野崇芳著／ナツメ社／B6判／302ページ／1,200円

QuickCの便利な活用本
手越義昭・新長輝則著／EI企画／井上書院（発売）／B5判／298ページ／2,600円

QuickCプログラミング
野口雄平・池田和徳・中島昌二著／ソフトバンク／A5判/368ページ／2,680円

QuickCプログラミングブック
塚本吉彦著／翔泳社／B5変判／312ページ／1,800円

QuickCプログラミング環境ーVer1.1/2.0対応版
黒瀬能聿・松尾俊彦著／啓学出版／B5判／180ページ／2,000円

RUN/C，Lattice C演習プログラムCランニングブック
横井与次郎著／ラジオ技術社／AB判／256ページ／2,580円

RUN/CVer.2対応C言語初恋レクチャ
庄司渉著／誠文堂新光社／A5判／172ページ／1,300円

RUN/Cで学ぶC言語2週間
R.トライスター著，浜本美紀訳／マグロウヒル出版／A5判／224ページ／1,957円

TURBO C 2.0ハンドブック（ハンディ・リファレンス36）
堀越一雄著／ナツメ社／B6判／440ページ／1,500円

TURBO C独学のすすめ
S.バーナップ著，渋谷恵津子訳，JTM企画監修／マグロウヒル出版／A5判／240ページ／2,369円

TURBO Cダイジェストガイド
塚本吉彦著／翔泳社／A5判／304ページ／1,600円

TURBO Cテクニカルガイド
服部昌博著／工学図書／B5判／224ページ／2,226ページ

TURBO Cプログラミング（Information & Computing第24巻）

MS-Cエラーメッセージ
Fatal error C 1126: 'identifier'： automatic allocation exceeds 'size'　関数のローカル変数用に割り当てられたスペースが指定された限度を超えた。

BOOK LIST◀

玉井浩著／サイエンス社／A5判／224ページ／1,860円
TURBO Cプログラミング入門
赤川勝矢著／啓学出版／A5判／228ページ／2,700円
TURBO Cプログラム・キット<上>
横井与次郎著／HBJ出版局／B5判／298ページ／2,680円
TURBO Cプログラム・キット<下>
横井与次郎著／HBJ出版局／B5判／324ページ／2,800円
TURBO Cマスター・マニュアル
斉藤孝・三和義秀著／HBJ出版局／B5判／162ページ／2,060円
TURBO Cユーティリティ・コレクション―誰でもわかるプログラム作成ガイド
アイクス企画著／HBJ出版局／B5判／201ページ／2,300円
TURBO C入門
荒実著／啓学出版／B5判／200ページ／2,500円
The C Trainer トレーニングマニュアル
アラン.R.フォイヤ著，玄光男訳／HBJ出版局／B5判／232ページ／2,369円
Turbo C Ver.2.0プログラミング
井上哲理・酒井恵光・佐藤隆著／ソフトバンク／A5判／416ページ／2,900円
Turbo C Ver.2.0プログラミングブック
田中正造著／翔泳社／B5変判／328ページ／2,000円
Turbo Cプログラミング入門
鷲地賢・阿部正浩著／アスキー／B5変判／288ページ／1,800円
Turbo C初歩の初歩講座
大竹政広・佐々木裕之著／日本文芸社／A5判／208ページ／1,600円
Turbo Debuggerハンドブック
寺尾敦著／新紀元社／A5判／263ページ／2,000円
TurboCの技法
S.R.デイビス著／工学社／B5判／432ページ／3,000円
TurboCの使い方
TurboC研究会著／工学社／A5判／440ページ／2,800円
CのかんどころーTURBO Cプログラミング入門（ラジオ技術選書194）
伊藤英治著／ラジオ技術社／A5判／311ページ／2,470円
すぐ使える Lattice C／パーソナル
堀越一雄・曽我泰介著／ナツメ社／菊判／216ページ／1,240円
はじめての Microsoft C
川崎盛美著／ナツメ社／菊判／221ページ／1,550円
はじめての TURBO Cプログラミング TURBO C入門
林晴比古著／ソフトバンク／A5判／372ページ／2,680円
よくわかる Datalight C OPTIMUM
植田一廣著／工学図書／A5判／180ページ／1,854円
よくわかる TURBO Cプログラミング
土屋勝・木全勝弘著／ナツメ社／菊判／304ページ／1,450円
ザ・TURBO C（NSライブラリ7）
戸川隼人・平島守著／サイエンス社／A5判／224ページ／1,860円
一夜漬 Lattice Cパーソナル―はじめてのC言語
JTM企画，浜本美紀著／ラジオ技術社／A5判／144ページ／1,340円
快適プログラミングのためのよくわかる TURBO C
アイクス企画著／HBJ出版局／B5判／162ページ／2,060円
改訂 TURBO Cプログラミング・ハンドブックII
宍倉幸則著／技術評論社／A5判／466ページ／2,400円
改訂 Ver.2.0版最新はじめての QuickC II
安田幸弘著／技術評論社／B5変判／282ページ／1,800円
改訂 Ver.2.0版最新はじめての TURBO C II
河西朝雄著／技術評論社／B5変判／240ページ／1,600円
改訂 Ver.2.0版 TURBO C初級プログラミング上
河西朝雄著／技術評論社／A5判／312ページ／2,200円
改訂 Ver.2.0版 TURBO C初級プログラミング下
河西朝雄著／技術評論社／A5判／450ページ／2,400円
自主トレCプログラミング
木下恂著／電気書院／A5判／250ページ／2,000円
自習 TURBO Cゲームで挑戦
TEAM TIPS編／技術評論社／A5判／262ページ／1,500円
徹底入門　新 Turbo　Cの世界―MS-DOSの基礎からCグラフィックステクニックまで
益岡和美著／山海堂／B5変判／184ページ／1,800円
入門 Lattice C
御手洗毅著／工学図書／A5判／208ページ／1,957円
入門 Microsoft C
御手洗毅著／工学図書／A5判／190ページ／1,957円
入門 TURBO C
御手洗毅著／工学図書／A5判／240ページ／2,060円
入門 TURBO C
荒実著／啓学出版／B5判／376ページ／3,610円

ちょっと視点を変えたCの本

ANSI C言語辞典
平林雅英著，椋田實・福富寛協力／技術評論社／B6変判／496ページ／2,100円
ANSI C言語大辞典
マーク・ウィリアムズ社編，坂本文・今泉貴史監訳／パーソナルメディア／A5判／600ページ／4,120円
Cの宝箱
A.ホラブ著，岩谷宏訳／工学社／B5判／472ページ／2,987円
Cプログラミングの落し穴（A-W/TOPPAN情報科学シリーズ）
A.コーニッグ著，中村明訳／トッパン／A5判／176ページ／2,300円
C言語用語ハンドブック
河西朝雄著／ナツメ社／B6判／320ページ／1,200円
教育用C言語の ABC
中山茂著／技報堂出版／A5判／260ページ／3,090円

パソコン博物館 IV

Z80成功物語の嚆矢
TRS80

タンディラディオシャック社TRS80

井戸禎光

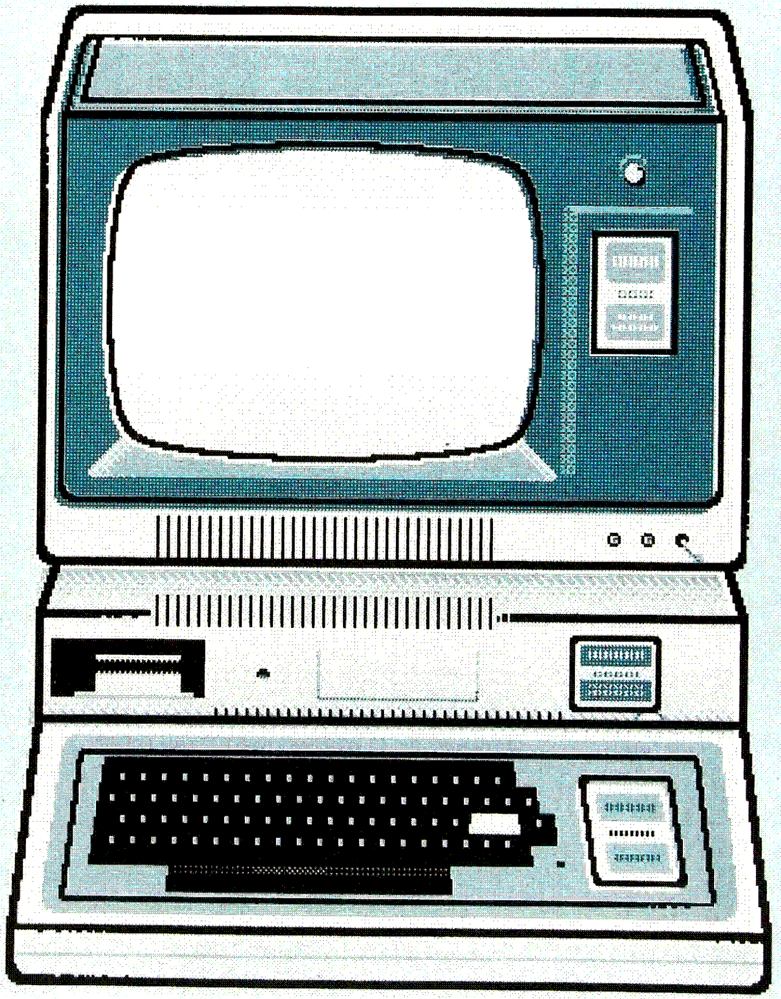


Illustration
TOSHIHITO ITO

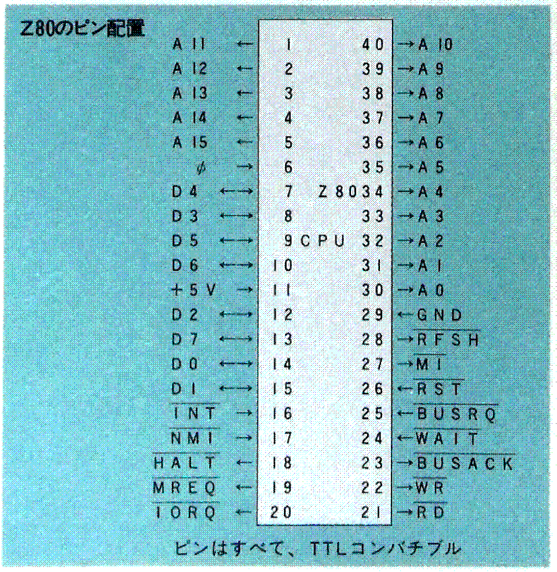
今回で，パソコン第2期黄金時代を築いた3機種（アップルII，コモドールPET，それに今回のTRS80）がでそろった．TRS80を売り出したタンディ社のことから述べよう．当時テキサス州のフォートワース市にあったタンディ社は，ラディオシャックという小売店網をもっていた．この小売店網は全米のラディオ，ステレオの流通市場の過半数を制するといわれていた．ステレオの小売店がパソコンを開発して売り出すなどということは，組織の壁，人材の教育など大手企業のもつ欠点を克服する必要があり，かなり難

しいことであったに違いない．しかしとにかく売り出したのである．

TRS80は1977年の12月に発売されており，価格は599ドル，本体のみでは395ドルであった．TRS80の成功した理由の大きな要素は，採用したCPUチップにある．そのZ80というチップは，1976年にザイログ社が発表した8200トランジスタを集積したチップである（ザイログ社は嶋正利氏，フェデリコ・ファジン氏らによって作られた会社であり，もとはインテル社の系統である）．このICが発売されてからパソコンが発表されるまで2年以下であるということは，注目に値するところである．

当時は8080が絶頂期であり，モトローラの6800をはじめとして，いくつかのICが開発されていた．一方，8080のほうは，ソフトウェアが十分に蓄積され相当使いやすくなってきていた．しかしハードウェアのほうは，ソフトウェアの充実に比例して要求が贅沢になり，新しい高度なICの出現を促していた．Z80はこのような時代の要請に応える形で現れた．

これは8080より格段に使いやすいCPUであり，たとえば，電源にしても8080は±5Vと+12Vの3電源であるのに対し，Z80は+5Vの1電源ですむし，D-RAMのリフレッシュ回路は別に作らなくてもすでに内部に入っていたりで，たいへんありがたかった．クロックにしても8080は2相クロックなので，8224というICを使うか外で対応する回路を組む必要があったし，メモリリード／ライトとかI/Oリード／ライトの信号を得るにも，8228などのICを必要とした．これらの点がZ80では改良されている．周辺のコントロール用ICはZ80用はけっこう価格が高かったが，8080用に作られたものが使えたので，これも便利であった．



Z80のピン配置

ピンはすべて、TTLコンパチブル

一方，ソフトウェアのほうでは，ストリング命令により256バイト，512バイトのブロック転送ができたり，16ビット演算などの新しい命令が増え，8080Aの78種の命令に対し158種の命令になった．そこで，究極の8ビットCPUといわれていた．しかし何といってもZ80の成功の原因は，8080のすべての命令を含む，アップワードコンパチビリティーのため，8080用にすでに開発されていたソフトウェア資産を継承しつつ性能向上が可能となったためである．このような特徴のあるICが発表されて，すぐに大手メーカーが飛びついたというのも興味深いことである．

オペレーティングシステムはまだ未成熟であり，各社各様のものを積んでいたが，タンディ社においてもTRSDOSというものを開発しており，やはり他社のものとは互換性がなかった．

さて，当時のパソコンマーケットの状況を調べると，テキサスインスツルメントのような大手とベンチャービジネスが混戦模様という様子であった．しかし実際は，大手とベンチャーではなく，量産・量販を得意とする家電系とアップルが残ることになる．実際コモドールとアタリの両社の努力により，価格はどんどん下がってゆき，大手メーカーはそれに耐えられず敗退していったのである．アタリ社の方はマネジメントの不在によって，企業の実態はいろいろな変遷を経験することになる．このような状況において，タンディ社は販売力にものをいわせてうまく生き延びることができそうであったが，重大な過ちを犯してしまった．すなわち，社外のソフトウェアメーカーがTRS80のためソフトを作ることを許さなかったこと，それにタンディラディオシャック社で他社で作ったソフトウェアを売らなかったことがあげられる．

タンディ社は1990年12月松下電器との合弁企業のプロジェクトを発表している．販売に強い企業同士の協力関係が，今後どのような発展の軌跡を描いていくか，たいへん興味のあるところである．

祐 安 重 夫

プログラマの道具学

第4回　キーボード

Illustration
Mikio Sasaki

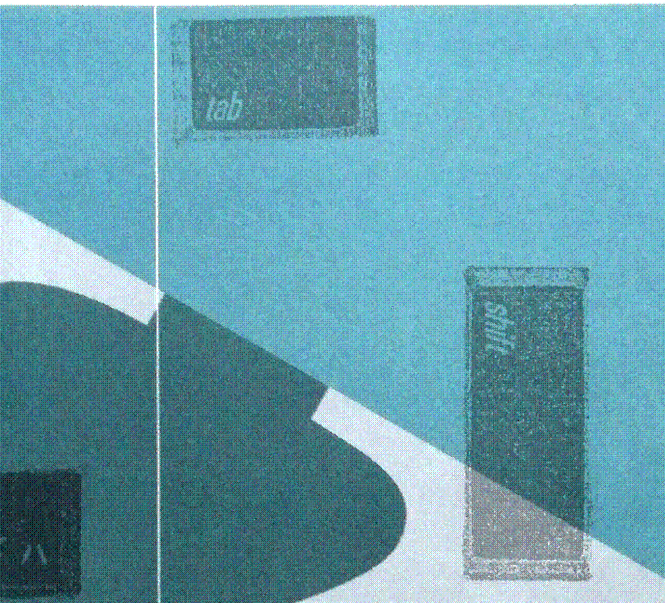
フレドリック・ブラウンのSF短編に，「エタオイン・シュルドゥル」Etaoin Shrdlu（『わが手の宇宙』早川書房）というのがある．地方のちいさな新聞社にやってきた不思議な男が，そこのライノタイプをかりて，秘密を要する印刷物の版下をつくるのだが，そのあとライノタイプが類似生命のようなものをもちはじめ，オペレータを無視して原稿のミスをそのまま植字したり，その原稿の内容を理解し，その内容に影響されるようになるという物語である．

ここからおこる騒動と結末は，未読の読者のために内緒にしておく．それよりもライノタイプである．ライノタイプとは活字鋳造機能と植字機能をかねそなえた欧文の活版組版機で，タイプライタとおなじくキーボードによって操作される．ETAOIN SHRDLUというのはキーボードの左端の縦2列の文字配列をあらわし，ライノタイプのためしうちにそのキーを順番にたたくことから，ブラウンの小説の中のライノタイプは，みずからETAOIN SHRDLUと名のりだす．

ETAOIN SHRDLUという奇妙な文字の配列は，じつは英文における文字を，出現頻度のたかい順にならべたものにほぼ相当しており，その意味ではライノタイプのキーボード配列は，合理的な理由にもとづいてきめられている．

ところで私たちがつかっているコンピュータのキーボードの文字配列は，タイプライタのキーボードの配列とほとんどおなじであり，欧米におけるタイプライタの歴史をそのままひきずっている．この配列は上から2段目の英字を左から右によんで，QWERTY配列などとよばれているが，これもまたETAOIN SHRDLUのように合理的な理由があってきめられたものなのだろうか．

コンピュータではキーボードと画面に表示



される文字とのあいだに，直接の機械的メカニズムは存在しないし，最近のタイプライタもほとんどがマイクロ・プロセッサをくみこんだ電子タイプライタになりつつある．しかし，現在でもキーボードの操作によって先端に活字が溶接された金属のバーがうごかされ，活字がリボンをたたく形式のタイプライタが，まったくなくなったわけではない．

昔風のタイプライタを操作したことがあればたいてい経験しているのが，キーボードを連続してたたいているときに，複数のバーがもつれてしまうという現象である．現在つかわれているタイプライタには，そのもつれをときほぐすためのメカニズムがくみこまれているし，それは致命的な欠陥ではない．

初期のタイプライタでは活字バーのもつれをおこさないために，印字速度がいちじるしくおそくなるといった問題がおこっていた．その後の材料の進歩やメカニズムの改良によって，このような致命的な欠陥は徐々にとりのぞかれてきたのだが，キーボードの配列だけは改良されることがなかった．

じつはQWERTY配列とは，英文の印字において活字バーのもつれが，できるだけおこりにくいように設計されたものだといわれている．その意味では合理的な配列といえるのかもしれないが，それがけっして使用する人間にとって合理的なものではなかったということが問題なのである．効率がよく使用する人間に負担をかけないという観点からすれば，QWERTY配列はかなりの欠点をもっている．

タイプライタの歴史の中で，キーボード配列の改良の提案はいくつもおこなわれてきた．その中でもオーガスト・ドヴォラクAugust Dvorakによるドヴォラク・キーボード(Dvorak Simplified Keyboard)は，習熟しやすく，ミスタイプがすくなく，タイプ速度がはやく，疲労度がすくないといった具合に，ほとんどの面でQWERTY配列よりすぐれていた．

にもかかわらずドヴォラク・キーボードは，現在でも普及してはいない．それはドヴォラクによる提案がおこなわれた時期が，大恐慌から第２次世界大戦にいたる時代であったこと，そして既存のタイプライタ製造業社や，すでにQWERTY配列に習熟したタイピストたちによる反発の結果といわれている．

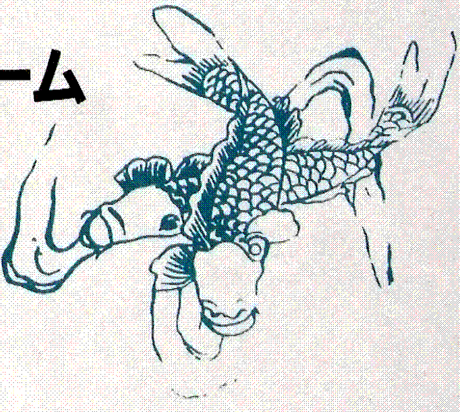
機械式タイプライタではキーボードの配列を変更するには，かなりのコストがかかった．活字バーの先の活字を，すべてつけかえる必要があったからである．しかし現在の電子タイプライタやコンピュータでは，プログラムの変更によってキーボード配列を，まったく別のものに変更することができる．

MS-DOSをつかったパーソナル・コンピュータの場合，デバイス・ドライバというプログラムをくみこむことで，ドヴォラク・キーボードにしてしまうことが可能である．機種によってはパブリック・ドメインというかたちで，そのようなプログラムが無償で公開されている場合がある．パーソナル・コンピュータが個人の多様性を容認する方向に機能している，ひとつの例といえるかもしれない．

プログラム遊工房

谷口典生

四乃巻
「マスターマインド」ゲーム



今回は，昔懐かしい『マスターマインド』ゲームをプログラムにしてみました．親の決めた石の色と位置を子が当てるゲームですが，コンピュータに親をやらせて１人で遊べるようになっています．
ワンポイントテクニックでは，マクロで定義した関数を使うときの注意をとりあげました．基本的なことですが，発見しにくいバグをひき起こすこともありますので，覚えておくとよいと思います．

リスト1

```
 1: /*
 2:  * milo.c        「マスターマインド」ゲーム
 3:  */
 4: 
 5: #include <time.h>
 6: 
 7: #define COLORS          5               /* 石の種類の数 */
 8: #define STONES          6               /* 並べる石の数 */
 9: #define MAX_TRY         20              /* 試行回数 */
10: #define BASE_X          10              /* X座標ベース */
11: #define BASE_Y          1               /* Y座標ベース */
12: 
13: #define gotoxy(x,y)     printf("¥x1b[%d;%dH",y,(x)*2+1)
14: 
15: int stone[MAX_TRY+1][STONES];
16: char ch_stone[COLORS][11] = {
17:         "¥x1b[34m●¥x1b[m",             /* 石その1 (青/丸型) */
18:         "¥x1b[31m■¥x1b[m",             /* 石その2 (赤/四角) */
19:         "¥x1b[32m▲¥x1b[m",             /* 石その3 (緑/三角) */
20:         "¥x1b[33m★¥x1b[m",             /* 石その4 (黄/星型) */
21:         "¥x1b[35m◆¥x1b[m" };           /* 石その5 (紫/菱形) */
22: 
23: main()
24: {
25:         int games = 0;                  /* 総ゲーム回数 */
26:         int average = 0;                /* 平均試行回数 */
27:         int result;                     /* ゲーム結果(試行回数) */
28: 
29:         init();                         /* 起動時イニシャライズ処理 */
30: 
31:         while(1) {
32:                 result = game();                /* 1 ゲーム実行 */
33:                 games++;
34:                 average = (average*(games-1) + result*100) / games;
35:                 printf("合計ゲーム数 = %d / ", games );
36:                 printf("平均回数 = %d.%d¥n",average/100, average%100 );
37:                 printf("もう一度やりますか? (Y/anykey)");
38:                 if(to_upper(getch())!='Y')
39:                         break;
40:         }
41: 
42:         deinit();                       /* 終了時イニシャライズ処理 */
43: }
44: 
45: generateanswer()                /* 解答となるパターンを作成する */
46: {
47:         int i;
48: 
49:         for(i=0;i<STONES;i++)
50:                 stone[0][i] = rand() % COLORS;
51: }
52: 
53: int game()              /* 1ゲーム実行する */
54: {
55:         int i;
56:         int result;                             /* ゲーム結果 */
57: 
58:         printf("¥x1b[2J");                      /* 画面クリア */
59:         generateanswer();                       /* 解答を作成 */
60:         gotoxy(BASE_X-5, BASE_Y);
61:         printf("解答      ");
62:         for(i=0;i<STONES;i++)
63:                 printf("？");
64:         result = try();
65:         showstone(0);                           /* 解答を表示 */
66:         if(result<0) {                          /* ギブアップだったか? */
67:                 gotoxy(0, BASE_Y+MAX_TRY+1);
68:                 printf("残念でした。ギブアップです。¥n");
69:                 return(MAX_TRY+1);              /* 最大試行回数+1 を返す */
70:         } else {
71:                 gotoxy(0, BASE_Y+result+1);
72:                 printf("おめでとうございます。");
73:                 printf("%d 回目でみごと正解です!!¥n",result);
74:                 return(result);                 /* かかった試行回数を返す */
75:         }
76: }
77: 
78: int try()               /* 試行を繰り返す */
79: {
80:         int turn;
81: 
82:         for(turn=1; turn<=MAX_TRY; turn++) {
83:                 gotoxy(BASE_X-5, BASE_Y+turn);
84:                 printf("%2d 回目", turn);       /* 何回目かを表示 */
85:                 selectstone(turn);              /* パターンの入力 */
86:                 if(checkstone(turn)==STONES)    /* すべての位置が合ったか? */
87:                         return(turn);
88:         }
89:         return(-1);                             /* ギブアップ */
```

プログラムの使い方

このプログラムはPC-9800シリーズ用で，Turbo C, Quick CおよびMS-Cに対応しています．リスト1を打ち込み，コンパイルして実行ファイルを作ってください．

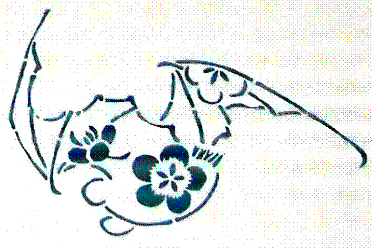
起動するには，たんに，

MILO

とすればゲームがはじまります．ゲームの遊び方については次項を参照してください．

1ゲーム終わると総ゲーム数と平均試行回数が表示され，もう1度やるかを聞いてくるので，ゲームを続けるには「Y」を，やめたい場合はそれ以外のキーを押します．

石の種類は5種類，並べる石の数は6になっていますが，それぞれの定数を変更することで難易度を調節することができます．最初，ルールがわからないうちは石の種類3，並べる石の数4くらいに減らしてやってみるとよいでしょう．石の種類を変更したときは，16～21行目でch_stoneに定義している文字列の数もそれに応じて変更してください．



```
}

int checkstone(turn)            /* 石の判定 */
        int turn;
{
        int tmp[STONES], mark[STONES];
        int i, j;
        int a_place=0, a_color=0;

        for(i=0; i<STONES; i++) {
                mark[i] = 1;                    /* 未判定フラブ */
                tmp[i] = stone[turn][i];        /* tmp[] にコピーする*/
                if(stone[0][i]==tmp[i]) {       /* 位置が合っているか? */
                        tmp[i] = -1;
                        mark[i] = 0;
                        a_place++;              /* 位置が合っている */
                }
        }

        for(i=0; i<STONES; i++)
                if(mark[i])                             /* 未判定か? */
                        for(j=0; j<STONES; j++)
                                if(stone[0][i]==tmp[j]) {
                                        tmp[j] = -1;
                                        a_color++;      /* 色が合っている */
                                        break;
                                }

        gotoxy(BASE_X+STONES+1, BASE_Y+turn);
        for(i=0;i<a_place;i++)  /* 位置が合っている石の数だけ黒丸を表示 */
                printf("●");
        for(i=0;i<a_color;i++)  /* 色が合っている石の数だけ白丸を表示 */
                printf("○");

        return(a_place);        /* 位置が合っている石の数を返す */
}

selectstone(turn)
        int turn;
{
        int i;
        int snum = 0;                   /* 石の位置番号 */
        int floop = 1;                  /* ループ続行フラグ */

        for(i=0;i<STONES;i++)                   /* 試行パターンの初期化 */
                stone[turn][i] = 0;

        while(floop) {
                showstone(turn);
                gotoxy(BASE_X+snum, BASE_Y+turn+1);
                printf("↑");                   /* ポインタ表示 */
                switch(to_upper(getch())) {
                        case '4' :              /* ポインタを左に移動 */
                        case 'H' :
                                snum--;
                                if(snum<0)
                                        snum = STONES-1;
                                break;
                        case '6' :              /* ポインタを右に移動 */
                        case 'L' :
                                snum++;
                                if(snum>=STONES)
                                        snum = 0;
                                break;
                        case '8' :              /* 石の色番号を増やす */
                        case 'K' :
                                stone[turn][snum]++;
                                if(stone[turn][snum]>=COLORS)
                                        stone[turn][snum] = 0;
                                break;
                        case '2' :              /* 石の色番号減らす */
                        case 'J' :
                                stone[turn][snum]--;
                                if(stone[turn][snum]<0)
                                        stone[turn][snum] = COLORS-1;
                                break;
                        case ' ' :              /* パターン決定 */
                        case 0x0d :
                                floop = 0;
                                break;
                        default :
                                printf("¥x07");
                }
                printf("¥x1b[2D  ");           /* ポインタ消去 */
        }
}

to_upper(c)                     /* 小文字->大文字変換 */
        int c;
{
        if(c>='a')
                return(c-'a'+'A');
        else
                return(c);
}

showstone(turn)
        int turn;
```

ゲームの遊び方

このゲーム，筆者の知り合いは『ミロ』という名前だったといい張るので，プログラム名は「milo.c」にしましたが，筆者は『マスターマインド』が先だと思っています（違ってたらごめんなさい）．最近では，『おむすび探偵団』というゲームが同じもののようです．

まず親（コンピュータ）が，6つの石を子（あなた）に見せずに並べます．子は，適当に石の並びを予想し，6つの石を並べて親に見せます．

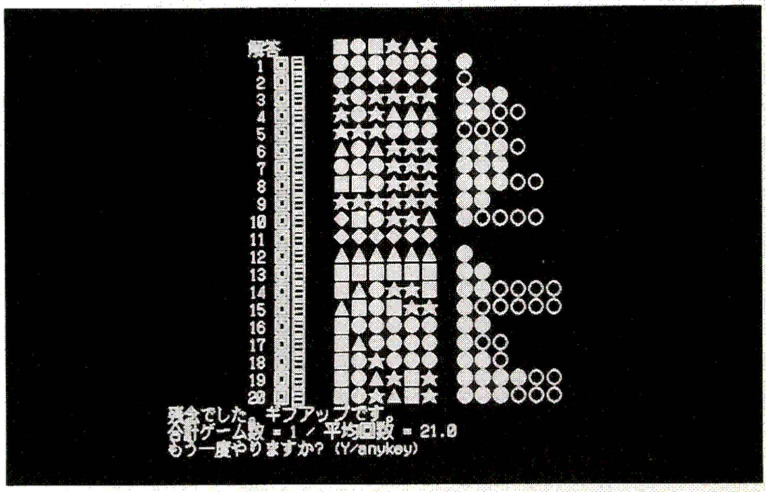
それに対して親は「場所も色もあっている石の数」を黒石の数で，「場所は違っているが，色はあっている石の数」を白石の数で子に示します．あくまでも「あっている石の数」だけを示し，実際にどの石があっているかは教えないわけです．子は，その情報を元に親の石を推理し，できるだけ少ない回数ですべての石を当てるのが目的です．

ゲームをはじめると，親の石が「?」で隠して表示され，子の石がその下に表示されます．カーソルの矢印をそれぞれの石の下に移動し，色を変更することができます．すべての石の色を決めると，親が判定結果を右に表示します．

使うキーは，カーソルの移動が「4」と「6」（または「H」と「L」），色の変更が「8」と「2」（または「K」と「J」），決定がスペースキーまたはリターンキーです．

試行を繰り返し，すべての石を当てる

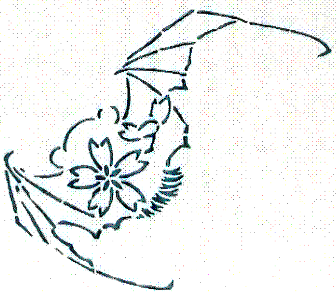
```
188: {
189:         int i;
190:         for(i=0;i<STONES;i++) {
191:                 gotoxy(BASE_X+i, BASE_Y+turn);
192:                 printf("%s", ch_stone[stone[turn][i]]);
193:         }
194: }
195:
196: init()
197: {
198:         time_t t;
199:
200:         time(&t);
201:         srand((unsigned int)t);          /* 乱数の初期化 */
202:
203:         printf("¥x1b[>5h¥x1b[>1h");
204:         /* カーソル表示停止/最下行をプログラムで使用 */
205: }
206:
207: deinit()
208: {
209:         printf("¥x1b[>5l¥x1b[>1l");
210:         /* カーソル表示開始/最下行をシステムで使用 */
211: }
212:
```



か，20回失敗すると1ゲーム終了です．

このゲームのコツは，うまく石を並べて，できるだけ少ない回数で多くの情報をひき出すことです．無駄な試行をしないように，計画的に石を並べるようにしましょう．たとえば，石の種類が5種類とわかっているのですから，4種類の石について使われている数がわかれば，残りの1種類がいくつ使われているかはわかるはずです．どの種類がいくつ使われているかを確実に知りたければ，「すべて青」「すべて赤」などを4回繰り返せば十分です．この方法は無駄が多いので(親の判定がすべて黒石になることがわかりきっている)，適当にほかの色の石を混ぜながら試行を繰り返す必要があるでしょう．

慣れれば，平均試行回数が10回を超えることはなくなるでしょう．より少ない試行回数で当てられるよう，いろいろ工夫してみてください．



プログラムの説明

7-11行

各種定数を定義します．

7行

何種類の石を使うか定義します．この値を大きくすれば，より多くの試行回数が必要になるでしょう．

8行

並べる石の数を定義します．この回数だけ試行しても答を当てられないときは，ギブアップとなります．ここでは20としてありますが，この値を増やす場合は，表示ルーチンも変更する必要があります．

10-11行

石を表示する座標のベースを定義します．

13行

gotoxy (x, y) という関数をマクロ定義します．表示位置をy行 (x*2+1) カラムに移動させる関数です．

15行

stone [0] [] には解答のパターンが入り，stone [1] [] ~stone [MAX_TRY] []に各試行パターンが入ります．

16-21行

ch_stone [c] [] には番号cの石の表示用データが入ります．ここでは，エスケープシーケンスを使用して石に色をつけています．

main()

31-40行

メインループ.

32行

game() を呼んで1ゲーム実行し, 結果 (かかった試行回数) をresultに代入します.

33-34行

ゲーム回数をインクリメントし, これまでの平均試行回数を求めます. averageにはつねに (平均試行回数×100) の値が入ります.

35-37行

合計ゲーム数, 平均試行回数, リプレイするかどうかのメッセージを表示します.

38-39行

入力したキーがY (またはy) でなければループから抜け出し, 終了します.

generateanswer()

解答となるパターンを作成し, stone[0] []に代入します. ここでは並べる石の数だけランダムに0～(COLORS-1)の番号を選んでいるだけですが, 同じ番号の石は一定の個数 (2個ないし3個) までしか使わないなどのルールを設けると, 戦略が変わっておもしろいかもしれません.

game()

58-63行

画面をクリアし, generateanswer() を呼んで解答となるパターンを作成します. 画面最上段に「解答」と「?」を並べる石の数だけ表示します.

64-75行

試行を繰り返すルーチンtry() を呼んだあと, 解答を表示します. try() からの戻り値が0より小さければギブアップなので, メッセージを表示し, (最大試行回数＋1) の値を返します. 戻り値が正なら正解のメッセージを表示し, かかった試行回数を返します.

try()

答を当てるか, ギブアップになるまで試行を繰り返します.

82-89行

最大試行回数だけループし, 正解が得られない場合は (-1) を返します.

83-87行

左側に何回目の試行かを表示し, パターンを入力させます. 判定ルーチンを呼び, すべての石の位置が当たっていればかかった試行回数を返します.

checkstone()

石の判定を行います. 解答のパターンと入力した試行パターンをくらべ, 位置があっている石の数と色だけあっている石の数を判定し, 位置があっている石の数を戻り値として返します.

99-107行

まず, それぞれの石について解答パターンの未判定フラグ (mark []) を立てておき, 同時に試行パターンをtmp[] にコピーします. 位置と色があっている場合には, 未判定フラグをクリアしてtmp [] の該当の石番号を (-1) にし, 位置があっているカウントを1増やします.

109-116行

もう1回, 未判定で残っている石についてtmp[] と照合していき, 同じ色の石があれば該当のtmp [] を (-1) にして, 色があっているカウントを1増やします.

118-122行

位置があっている石の数だけ黒丸を, 色があっている石の数だけ白丸を表示します (画面では, 白と黒が逆になるので注意してください).

selectstone()

134-135行

試行パターンを格納する配列を0で初期化します.

138-140行

試行する石のパターンを画面に表示し, 色を変更するためのポインタをその石の下に表示します.

141行

1文字キー入力し, それぞれの入力に従った処理を行います.

142-147行

「4」または「H」のキーが押された場合は, ポインタを1つ左に動かします. ポインタが左端のときには右端に移動します.

148-153行

「6」または「L」のキーが押された場合は, ポインタを1つ右に動かします. ポインタが右端のときには左端に移動します.

154-159行

「8」または「K」のキーが押された場合は, ポインタが指し示す石の色番号を1つ増やします. 色番号が最大値の場合は0に戻します.

160-165行

「2」または「J」のキーが押された場合は, ポインタが指し示す石の色番号を1つ減らします. 色番号が0の場合は最大値に戻します.

166-169行

スペースまたはリターンキーが押された場合は, ループを抜け出します.

170-171行

それ以外のキーが押された場合は, ベルを鳴らします.

172-174行

ポインタを消去し, whileループの先頭へ戻ります.

❖to＿upper()

英小文字を大文字に変換します．ワンポイントテクニックの項参照．

❖showstone()

石を画面に表示します．引数が０の場合は解答のパターンが表示されます．

❖init()

乱数を初期化し，エスケープシーケンスによってカーソル表示を停止し，ファンクションキー表示エリアをプログラムで使えるようにします．

❖deinit()

カーソルを表示させ，ファンクションキー表示エリアを元に戻します．

図１　マクロ関数を使ったプログラム

```
#define gotoxy(x,y) printf("¥x1b[%d;%dH",y,x*2+1)
                 .
                 .
       gotoxy( BASE_X+10, BASE_Y );
```

図２　前処理によって展開された形

```
printf("¥x1b[%d;%dH",BASE_Y,BASE_X+10*2+1)
```

図３　正しく展開された形

```
printf("¥x1b[%d;%dH",BASE_Y,(BASE_X+10)*2+1)
```

図４　副作用を起こすプログラム例

```
#define abs(x) ((x)<0 ? -(x) : (x))

        i = -10;
        j = abs(i+=5);
```

図５　図４が展開された形

```
i = -10;
j = ((i+=5)<0 ? -(i+=5) : (i+=5));
```

ワンポイントテクニック

マクロ形式の関数と副作用

Ｃのプログラムでは，マクロとして定義した関数（マクロ関数）を使うことがよくあります．今回のプログラムでも，13行目で「#define gotoxy (x, y)　printf ("¥x1b [%d;%dH", y, (x) *2+1)」とマクロ関数を使用しています．このように定義しておけば，以後プログラムの中で，「gotoxy (x, y) ;」などのように関数の呼び出しと同様に使うことができるのです．

ただし，マクロ関数はコンパイラの前処理で置換によって展開しているだけなので，いくつか注意しておかなければいけないことがあります．

たとえば，上のgotoxy () の定義の中で「(x) *2+1」と仮引数を括弧でくくっていますが，これを「x*2+1」と書くとなにか不都合があるのでしょうか．これは，マクロが展開された形を考えてみればわかるでしょう．図１のプログラムは，図２のように展開されますが，よく見てみると「BASE_X+10*2+1」という部分が望ましくない形になっています．この部分は図３のように展開されることを期待していたのですから，仮引数をくくっていた括弧をとってはいけないことがわかると思います．

もう１つ，マクロ関数には「副作用」と呼ばれる問題があります．たとえば，図４のようなプログラムを考えてみましょう．abs () は整数の絶対値を求める関数をマクロ関数として作ったものですが，このような使い方をすると期待した結果が得られません．ｉの値は (-5) に，ｊの値は５になることを期待していたのに，どちらも０になってしまいます．図５のように展開された形を見ればわかるように，(i+=5) という式が２回評価されてしまうのが原因です．

今回のプログラムの中の「to_upper ()」という関数は，同じものがTurbo C，Quick CおよびMS-Cには「toupper ()」という関数としてライブラリに存在しています．もちろんそれを使ってもよかったのですが，Quick CやMS-Cの場合，ctype. hの中でこの関数はマクロで定義されているのです．今回は，ctype. hをインクルードしていないので関係ありませんが，「toupper (getch ())」のような使い方をしていると副作用が恐いので，筆者は安心のため別の関数として作る習慣がついてしまっているのです．

普通の関数にくらべて，呼び出しのためのオーバーヘッドがなく便利なマクロ関数ですが，副作用などをつねに頭に入れて使うようにしましょう（蛇足ですが，C++にはマクロ関数の信頼性をあげるため，インライン関数というものが用意されています）．

MS-DOS TOOL LABO

新連載
MS-DOSツール・ラボ

第1回 ディレクトリ表示ツール L.EXE

加野島英渡

今月号より開始するこの「MS-DOSツール・ラボ」では，マクロアセンブラを用いてMS-DOS上で動作する簡易Software Toolsといったプログラムを紹介していく．掲載するソースは短いが，その中にはマクロアセンブラの基本テクニックや，MS-DOSのシステムコールの利用法などがギッシリとつまっている．今回紹介するのは，MS-DOSのDIRコマンドとよく似た動作をするディレクトリ表示ツールである．

アセンブラでツールを

アセンブラでツールを書けとの仰せである．それもただ書くだけではおさまりがつきそうもない．書いたうえで読者の皆様にお見せして，恐れおおいことに講釈までせよとのお達しが出ている．う～む．

昨今ではツールにかぎらず，たいていのプログラムはなんらかの高水準言語で書かれているようだ．最近の高水準言語は開発環境が整備され，また非常に性能がよくなってきているので，昔のように苦労してプログラムすべてをアセンブラで開発する必要があまりなくなってきている．もちろんMS-DOSにはあいかわらずメモリの壁があり，デバイスドライバなどのメモリに常駐するプログラムは，できるだけサイズを小さくするためにアセンブラでかかれることが多い．しかし，これも工夫しだいでCなどの高水準言語で対応することができる．

では，アセンブラは不要か……というと，これがそうでもないのである．高速性，省メモリなどの特性からまだまだアセンブラの出番はなくなりそうもない．とくにCのようなアセンブラに近い記述のできる言語を使っている人は，アセンブラを知っているほうが効率のいいコーディングができる．一部のプログラマの間では，「書きながらアセンブラのソースが見えてくるようなやつでないとCなどまともに使えない」などといわれているくらいである．これには少々異論もあろうが，私がCを使いはじめたころにも，「このポインタの動きはどうも変だ」などと思うと，すぐにアセンブラのソースを吐き出させて確認したものである．やはり，知らないよりも知っているほうがいいよ

ワンラインNEWS
データイーストはインテルジャパンと共同でCD-ROMを利用した業務用ゲーム機を開発した．インテル社が開発した画像圧縮技術DVIを応用している．

図1　ディスク上のディレクトリ領域

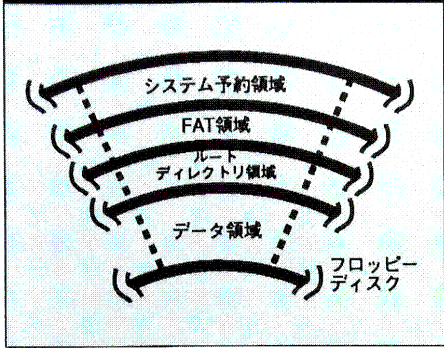
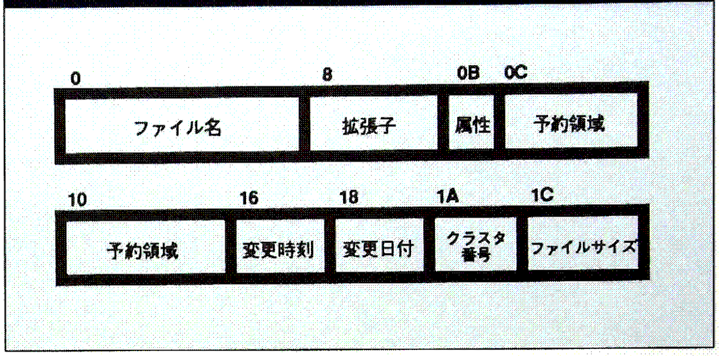


図2　ディレクトリエントリの構造



うだ。

というわけで，今月からしばらくおつきあいをお願いすることになった。毎月1本ずつ，なにかしらツールのようなものをアセンブラで書いては，なにがしかの解説をさせていただく。そのときどきで，マクロの話であったり，DOSのファンクションリクエストの使い方であったり，気分に応じて冗談であったり愚痴であったり諸行無常であったりもする。なんだかわからない。

わからないけれども，なにか1つくらいは「あぁ，これはためになった」と感じていただけるように心がけるので，ひとつよろしくお願いしたい。感想などは，編集部まで送っていただければ，たぶん回送してもらえると思う。苦情，叱責，ご意見，有価証券などなんでもありがたく頂戴する。

ディレクトリの構造

そういえば，CP/Mをはじめて使ったとき，真っ先に作ったのがこのディレクトリ表示ツールだったなぁ。わりと簡単に作れることと，標準でついているディレクトリ表示コマンドがファイルのトータルサイズを表示してくれないということが，オリジナルツールを作成する動機だった。まぁ，ちょっと手軽に「自作のコマンド」というのを作ってみたかったわけだ。

さて，MS-DOSのディスク上には，ファイル本体を書き込むデータ領域，ファイルのread/writeを管理するためのディレクトリ領域，およびFAT領域，さらにMS-DOSを起動するためのブートプログラムが書き込まれているシステム予約領域がある。

これらのうち，ディレクトリ領域には，ルートディレクトリに存在するすべてのファイルの情報――ファイル名，ファイルサイズ，ファイル作成日時，クラスタ番号，etc.――が収められている。この部分をディレクトリエントリと呼び，その大きさは32バイトである。サブディレクトリはディスクのデータ領域にほかのファイルとともに存在し，サブディレクトリ内のファイル情報をやはりディレクトリエントリの形で収めている。

ディレクトリエントリの構造は，図2のとおりである。このディレクトリエントリによって，MS-DOSの階層ディレクトリ構造が作られているわけだ（図3）。

つまり，ディレクトリを表示するとは，ディレクトリエントリの内容を読

図3　階層ディレクトリの構造

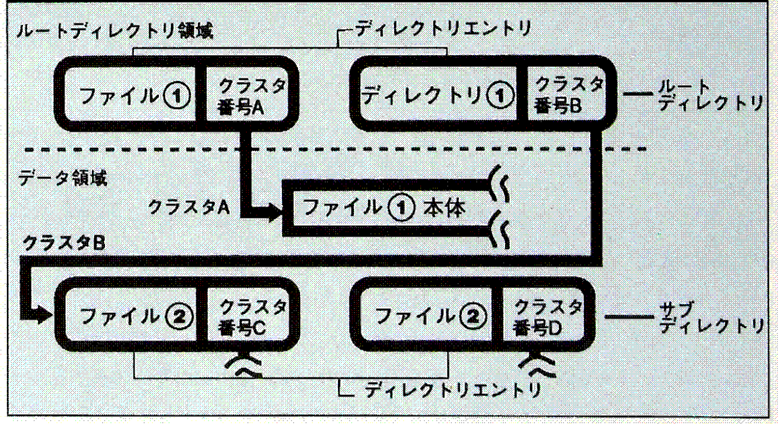


図4　ファンクション4EH　一致するファイルの検索

呼び出し手順
AH ← 4EHをセット
CX ← 検索するファイルの属性情報をセット
DS:DX ← ASCIZ文字列形式のファイル名の先頭アドレスをセット

戻り値
CY → 検索成功ならクリア，エラーならセットされ，AXにエラーコード
　AX → 02H　パス名が無効，該当ファイルが存在しない
　AX → 12H　これ以上ファイルが存在しない

ワンラインNEWS
理経は米E&S社製の三次元グラフィックWS「ESV3sc」を発売した。価格は6,138,000円から設定し，従来機に比べ最大24%の値下げをした。

図5　ASCIZ文字列形式の例

```
ASCIZ "ABC.DEF" = |41H|42H|43H|2EH|44H|45H|46H|00H|
                    'A' 'B' 'C' '.' 'D' 'E' 'F'
```

図6　ファイルの属性情報

7	6	5	4	3	2	1	0	
未使用	未使用	アーカイブ	サブディレクトリ	ボリュームラベル	システムファイル	不可視ファイル	リードオンリーファイル	
0	0	1	0	0	0	0	0	アーカイブ属性
0	0	0	1	0	0	0	0	サブディレクトリ
0	0	0	0	1	0	0	0	ボリュームラベル
0	0	0	0	0	1	0	0	システムファイル
0	0	0	0	0	0	1	0	隠しファイル
0	0	0	0	0	0	0	1	リードオンリーファイル

図7　DTAにセットされる情報のフォーマット

```
DTA + 00H～14H : 「次のファイルの検索」のために使われる情報
DTA +      15H : 発見されたファイルの属性
DTA + 16H～17H : このファイルに最後に書き込みが行われた時刻
DTA + 18H～19H : このファイルに最後に書き込みが行われた日付
DTA + 1AH～1BH : ファイルサイズ（下位ワード）
DTA + 1CH～1DH : ファイルサイズ（上位ワード）
DTA + 1EH～2AH : 発見されたファイル名（ASCIZ文字列形式）

      時刻データの詳細                  日付データの詳細
| HHHHH | MMMMMM | SSSSS |        | YYYYYYY | MMMM | DDDDD |

H : 時                            Y : 年 - 1980
M : 分                            M : 月
S : 秒 / 2                        D : 日
```

図8　ファンクション4FH　次に一致するファイルの検索

```
呼び出し手順
  AH ← 4FHをセット
戻り値
  CY → 検索成功ならクリア、エラーならセットされ、AXにエラーコード
       AX → 12H　これ以上ファイルが存在しない
```

み込んで画面（標準出力）に出力するという操作を意味していることになる．

実際にディレクトリを表示するための手順は簡単だ．検索するファイル名をコマンドラインオプションから得て，そのファイルが存在すればそのディレクトリエントリ内のファイル情報を表示すればいい．ただし，ファイル名にワイルドカードが含まれる場合があるので，発見されるファイル名は1つとは限らない．このへんがむずかしそうだが，MS-DOSにはワイルドカードも受け付けるファイル検索のためのファンクションリクエストが用意されているので，これを使えばいい．

ファンクションリクエスト

MS-DOSは，その内部の機能をシステムコールという形で公開している．プログラマは，自分のプログラムの中から，自由にこれらの機能を呼び出して利用することができる．システムコールの中でも，とくにINT 21Hによるものをファンクションリクエストと呼んでいる．

ファンクションリクエストは，アセンブラプログラムの中で，次のようにして実現する．

1．コールしようとするファンクションナンバーをAHレジスタにセットする．

2．実行しようとしているファンクションが必要とするパラメータを指定されたレジスタにセットする．

3．ソフトウェア割り込みINT 21Hを実行する．

4．結果はレジスタにセットされるか，または動作として処理される．

■最初のファイルの検索

ここでは，ファイル検索のためのファンクションリクエストとしてファンクション4EHを用いる（図4）．

ここでファイル名の先頭アドレスのセットに利用しているASCIZ文字列形式とは，00Hで終了する不定長の文字列形式で，C言語ではおなじみのものだ（図5）．ファイル名にはワイルドカードを使用することもできる．

CXにセットするファイルの属性情報とは，ディレクトリエントリに含まれる属性情報と同じで，図6のようなビット情報である．

通常のファイル以外に，ビットの立っている属性のファイルが検索対象に追加される．上記の属性をもつファイルのみを検索する方法はないので，たとえばサブディレクトリを検索する場合はビット4を立ててファイル検索し，発見されたファイルがサブディレクトリであるかどうかを，あらためて検査しなくてはならない．

発見されたファイルに関する情報は，DTA(Disk Transfer Address)と呼ばれるメモリ上のエリアに格納される．このエリアのアドレスはファンクションリクエストによって設定することができる．DTAにセットされる情報のフォーマットは図7のとおり．

■次のファイルの検索

次のファイルの検索には，ファンクション4FHを用いる（図8）．

ワンラインNEWS
日本テクサはソフト会社のメッツと業務提携．日本テクサのHDDにメッツのソフトを付属して販売する一方，メッツは日本テクサの開発したソフトを販売する．

このファンクションは必ず「最初のファイルの検索」に続いて使用する。以下CYがセットされるまでこのファンクションを続けて呼び出せば，指定した全ファイルの検索が終了する。なお，DTAは「最初のファイルの検索」で使用したものをそのまま使用しなければならない。

また，今回のプログラムでは，ファンクション4EH，4FH以外にも，図9～11のようなファンクションリクエストを利用している。

プログラムの特徴

今回掲載したのは，非常に簡単なディレクトリ表示プログラムである。設計にあたっては，できるだけDOSコマンドであるDIRと違和感がないように，使い勝手をあわせることを考えた。指定されたファイル名がサブディレクトリであった場合は，その下にあるファイルをすべて表示する，拡張子が省略された場合は'.*'が指定されたものとして扱う，などの点である。

また，ファイル属性もいっしょに表示するようにした。ちょっと画面がさびしいので，エスケープシーケンスを使って，各ビットごとに違う色をつけている。画面の出力をリダイレクトすると，このエスケープシーケンスもいっしょにファイルに残ってしまうので，エスケープシーケンスだけは標準エラーハンドルで出力している。COMMAND.COMの仕様ではエラー出力はリダイレクトされないからだ。

ファイルサイズは3桁ごとのカンマ区切りで見やすくし，最後に合計サイズを表示するようにしてみた。ただし，発見ファイルが1個の場合はうっとうしいので表示しない。なかなか芸が細かいのである。

プログラムの流れ

大まかな流れは，以下のようになっている。細かな説明は，プログラムにコメントを書いておいたのでそちらを参照していただきたい。

(1) 初期化

TPAにセットされているコマンドラインオプションを自分のデータセグメント内にコピーし，DTAも同じく自分のデータセグメント内にセットする。TPAをそのまま使うこともできるのだが，自分のデータセグメント内にもってきたほうがあとの処理が楽になる。あわせて，内部で使う変数類を初期化しておく。

(2) ファイル検索と表示

図9　ファンクション1AH　DTAアドレスの設定

呼び出し手順
AH ← 1AHをセット
DS:DX → ＤＴＡアドレスをセット
戻り値
なし

図10　ファンクション40H　デバイス・ファイルへの出力

呼び出し手順
AH ← 40Hをセット
BX ← ファイルハンドルをセット
CX ← 出力するバイト数をセット
DS:DX ← 出力データの先頭アドレスをセット
戻り値
CY → 検索成功ならクリア、AXには出力されたバイト数。
　　エラーならセットされ、AXにエラーコード
　　AX → 05H　アクセスは否定された
　　AX → 06H　無効なファイルハンドル

図11　ファンクション4CH　プロセスの終了

呼び出し手順
AH ← 4CHをセット
AL ← リターンコードをセット
戻り値
なし

リスト1

```
;;;;;;;;;;;;;;;;;;;;;;;;;;
;          console.h     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; ========================= マクロ定義 =============
;
PRINTM          macro   handle,arg
                local   prtdt
        push    bx
        mov     bx,handle
        call    printmes
        dw      prtdt - $ - 2
        db      arg
prtdt:
        pop     bx
ENDM
```

リスト2

```
;;;;;;;;;;;;;;;;;;;;
;     dosfunc.h    ;
;;;;;;;;;;;;;;;;;;;;
;
H_STDOUT        equ     1               ; 標準出力ハンドル
H_ERROR         equ     2               ; 標準エラー出力ハンドル
;
NORMAL_END      equ     0               ; プロセス正常終了コード
;
AT_A            equ     00100000b       ; アーカイブ属性
AT_D            equ     00010000b       ; サブディレクトリ
AT_V            equ     00001000b       ; ボリュームラベル
AT_S            equ     00000100b       ; システムファイル
AT_H            equ     00000010b       ; 隠しファイル
AT_R            equ     00000001b       ; リードオンリーファイル
;
ATTR_ALL        equ     AT_A + AT_D + AT_V + AT_S + AT_H + AT_R
;
; === ＩＮＴ－２１Ｈ機能番号 ===
;
DOS_SETDTA      equ     1ah             ; ＤＴＡアドレス設定
DOS_HWRITE      equ     40h             ; ファイル／デバイスへの出力
DOS_TERMINATE   equ     4ch             ; プロセス終了
DOS_FINDFIRST   equ     4eh             ; 最初のファイルの検索
DOS_FINDNEXT    equ     4fh             ; つぎのファイルの検索
;
```

コマンドラインオプションで指定されたファイル名を検索し，発見したものを順次表示する．本プログラムのメインロジックである．

(3)ファイル合計サイズの表示と終了

発見したファイルのサイズは内部の変数に加算してあるので，最後に表示してプログラムを終了する．

これらのプログラムは，以下のようにしてアセンブル・リンクすればよい．

```
A> masm l
A> masm console
A> link l+console
```

今月のお勧め

“num_str_sub”と，その上位ルーチンがお勧めである．このルーチンでは，AXBXに格納されているダブルワードの数値を文字列に変換している．

ワード値の範囲ならもっと簡単にできるのだが，ファイルサイズなどワード値に納まらないものも処理しなければならないので，ダブルワードの処理が必要になった．サブルーチン“num_str_sub”の動作は，AXBXをCXで割った余りをASCII文字に変換して，指定アドレスに格納するというものである．格納アドレスはDS：DIで指定し，格納後はDIをデクリメントして返す．戻ったときにAXBXには商が入っているので，これがゼロになるまで連続してこのサブルーチンを呼び出せば変換が終了する．

もちろん，桁数があらかじめわかっている場合はその桁数だけ繰り返せばいいし，余分に繰り返せば先頭に‘0’がパディングされていくので，桁あわせも可能である．

上位ルーチンの1つである“num_to_str”は，3桁ごとのカンマ区切り機能をもっている．この処理はそうむずかしいものではないのだが，はじめて作るときはけっこうてまどるものだ．よかったらこのまま抜き出して使ってほしい．なお，これにかぎらず各処理はできるだけ部品として使いやすいようにレジスタ保存をしている．ムズムズする向きもあろうが，ガマンしてもらいたい．

今月の反省

コマンドラインオプションの取得は，本来ならばちゃんと解析して先頭アドレスを配列にでも入れておくべきである．今回はちょっとこのへんをサボってしまったので，次回ではきちんとこの処理をやってみたい．

また，ファイル名の解析もちゃんとやっていないので，拡張子省略の検査がファイル名に漢字を使われるとしくじる可能性がある．これも，次回以降でファイル名をちゃんと解析する方法を書くことにしよう．どうも「次のバージョンで直します」といって逃げるという悪い習性が出てしまったようで申しわけない．ペコペコ

参考文献

●MS-DOSエンサイクロペディア　Vol.1　アスキー出版局

●Cの宝箱　アレン・ホラブ著(岩谷宏訳)　工学社

●ザ8086ブック　ラッセル・レクター／ジョージ・アレクシー共著(吉川敏則訳)　秋葉出版

リスト3

```
TITLE   console.asm コンソール入出力処理
PAGE    60,128
include         console.h
include         dosfunc.h
_TEXT   SEGMENT  WORD PUBLIC 'CODE'
_TEXT   ENDS
_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS
_STACK  SEGMENT STACK
_STACK  ENDS
DGROUP  GROUP   _DATA,_STACK
        ASSUME DS: DGROUP, SS: DGROUP
_TEXT      SEGMENT
        ASSUME  CS: _TEXT
;
public          printmes,putchar
public          color_normal,color_red,color_green,color_yellow
public          color_blue,color_magenta,color_cyan,color_white
;
; ----------------- 標準出力への出力機能
;
; === 戻り番地のつぎのアドレスからのデータを出力 ===
;
printmes        proc
        push    ax
        push    cx
        push    dx
        push    ds
        push    si
        push    bp
        mov     bp,sp
;
        mov     si,[bp + 12]    ; 戻りアドレスを取得して
        mov     cx,cs:[si]      ; そこから表示データのバイト数を得る.
        add     si,2            ; 表示データアドレスを指して
;
        mov     dx,si           ; それをＤＸにも設定し
        add     si,cx           ; 最後のデータのつぎを指して
        mov     [bp + 12],si    ; それを戻り番地とする.
;
        mov     ax,cs           ; コードセグメント上のデータなので、
        mov     ds,ax           ; それをＤＳにセットし（ＤＸはセットずみ）
        mov     ah,DOS_HWRITE   ; 標準出力に
        int     21h             ; 出力する.
;
        pop     bp
        pop     si
        pop     ds
        pop     dx
        pop     cx
        pop     ax
        ret                     ; このとき戻りアドレスは更新されている.
printmes        endp
;
; === 一文字出力 ===
;
putchar         proc
        push    bx
        push    cx
        push    dx
        push    ds
;
        push    ax              ; 表示データをＰＵＳＨする.
;
        mov     dx,sp           ; データはスタックの上にある.
        mov     ax,ss
        mov     ds,ax
        mov     cx,1
        mov     bx,H_STDOUT     ; 本当はもっと簡単なファンクションが
```


ワンラインNEWS
松下電器産業はワープロ検定のレッスン用ソフトを内蔵したワープロを発売した．ブラインドタッチなどが独学できる．

```
        mov     ah,DOS_HWRITE   ; あるのだが、出力を統一したかったので.
        int     21h
;
        pop     ax
;
        pop     ds
        pop     dx
        pop     cx
        pop     bx
        ret
putchar         endp
;
; ──────────── エスケープシーケンスによる表示色変更処理.
;               エラーハンドルに出力することで、リダイレ
;               クト時にファイルにエスケープシーケンスが
;               残ってしまうことを避けている. もし残した
;               いならば、"H_ERROR"を"H_STDOUT"に変えること.
;
color_normal    proc
        PRINTM  H_ERROR,<1bh,'[0m'>
        ret
color_normal    endp

color_red       proc
        PRINTM  H_ERROR,<1bh,'[31m'>
        ret
color_red       endp

color_green     proc
        PRINTM  H_ERROR,<1bh,'[32m'>
        ret
color_green     endp

color_yellow    proc
        PRINTM  H_ERROR,<1bh,'[33m'>
        ret
color_yellow    endp

color_blue      proc
        PRINTM  H_ERROR,<1bh,'[34m'>
        ret
color_blue      endp

color_magenta   proc
        PRINTM  H_ERROR,<1bh,'[35m'>
        ret
color_magenta   endp

color_cyan      proc
        PRINTM  H_ERROR,<1bh,'[36m'>
        ret
color_cyan      endp

color_white     proc
        PRINTM  H_ERROR,<1bh,'[37m'>
        ret
color_white     endp

_TEXT   ENDS

        END
```

リスト 4

```
TITLE   Command L.EXE

PAGE    60,128

include         console.h
include         dosfunc.h

_TEXT   SEGMENT  WORD PUBLIC 'CODE'
_TEXT   ENDS

_DATA   SEGMENT  WORD PUBLIC 'DATA'
_DATA   ENDS

_STACK  SEGMENT STACK
_STACK  ENDS

DGROUP  GROUP   _DATA,_STACK
        ASSUME DS: DGROUP, SS: DGROUP
;
; ====================== 定数宣言 ===========
;
SUBWILD         equu    1               ; 0にすると、サブディレクトリ名を
                                        ; 指定した場合にその下の全ファイルを
                                        ; 表示する機能を抑止できる.
;
PSP_PARAM       equ     80h             ; PSPのコマンドラインオプション
                                        ; のオフセット.
;
; ===============================================
;

_TEXT       SEGMENT
        ASSUME  CS: _TEXT
;
extrn           printmes:near,putchar:near
extrn           color_normal:near,color_red:near,color_green:near
extrn           color_yellow:near,color_blue:near,color_magenta:near
extrn           color_cyan:near,color_white:near
;
; ──────────── イニシャライズ
;
; 入力レジスタ  なし
; 出力レジスタ  なし
; 破壊レジスタ  AX,CX,DX,SI,DI,ES,DS
;
ls_initialize   proc
        mov     ax,DGROUP
        mov     es,ax
        mov     si,PSP_PARAM    ; コマンドラインオプションの
        lodsb                   ; バイト数を取得する.
        mov     cl,al
        mov     ch,0
        sub     si,2
        add     cx,2
skipsw:                         ; 空白を取り除く処理
        inc     si
        dec     cx
skipsb:
        inc     si
        dec     cx
        cmp     byte ptr [si],' '
        jz      skipsb
        cmp     word ptr [si],'  '
        jz      skipsw
;
        mov     es:[paramc],cx  ; 空白を除いたあとのバイト数を保存する.
        mov     di,offset param ; 自分のデータセグメント内の領域に
rep     movsb                   ; コマンドラインオプションをコピーする.
        mov     ax,es
        mov     ds,ax           ; DSも同じデータセグメントを指すようにし、
        mov     byte ptr [di],0 ; コマンドラインオプションを終端する.
;
        mov     dx,offset dta
        mov     ah,DOS_SETDTA
        int     21h             ; DTAも自分のデータセグメント内にセット.
;
        mov     ax,0            ; ファイルサイズ合計とファイル数の初期化.
        mov     word ptr [size_total],ax
        mov     word ptr [size_total + 2],ax
        mov     [file_count],ax
        ret
ls_initialize   endp
;
; ──────────── コマンドラインのワイルドカード指定文字を検査する
;
; 入力レジスタ  なし
; 出力レジスタ  Z:1=ワイルドカードあり,Z:0=なし
;
check_wildcard  proc
        push    ax
        push    cx
        push    si
        mov     cx,[paramc]     ; コマンドラインオプションのバイト数だけ検査.
        mov     si,offset param
_chkw_lp:
        lodsb
        cmp     al,'*'
        jz      _chkwe
        cmp     al,'?'
        jz      _chkwe
        loop    _chkw_lp
        cmp     cx,-1           ; ゼロフラグを落とす. (NZにする)
_chkwe:
        pop     si
        pop     cx
        pop     ax
        ret
check_wildcard  endp
;
; ──────────── ファイル名記述省略の処理
;
; 入力レジスタ  DS:DX=ファイル名先頭アドレス
; 出力レジスタ  なし
;
check_omission  proc
        push    ax
        push    cx
        push    di
;
        mov     di,dx
        cmp     byte ptr[di],0
        jz      _cda_set1       ; オプションなしは'*.*'とする.
_cda1:
        add     di,[paramc]
        mov     al,[di - 1]     ; オプションの末尾が
        cmp     al,'¥'          ; '¥'か
        jz      _cda_set1
        cmp     al,':'          ; ':'だったら
        jz      _cda_set1       ; '*.*'が指定されたものとする.
;
if      SUBWILD ; ===========================================
        call    check_wildcard
        jz      _cda_ext        ; ワイルドカード指定がなくて
        mov     cx,AT_D         ; サブディレクトリ属性のファイルが
        mov     ah,DOS_FINDFIRST
        int     21h
        jc      _cda_ext        ; 見つからなかったら終わり
        test    byte ptr[dta_attr],AT_D
        jnz     _cda_set0       ; 見つかったら'*.*'を追加.
endif                   ; ===========================================
;
_cda_ext:                       ; 拡張子省略の検査.
        mov     cx,[paramc]
        push    di              ; DIは終端文字'00H'を指している
_cda_ext_lp:
        dec     di
```

ワンラインNEWS
日立製作所は世界初，4MビットスタティックRAM「HM628512」シリーズを製品化した．ハーフミクロン技術により高速・低消費電力を実現．

```
        mov     al,[di]
        cmp     al,'Y'          ; 'Y'か
        jz      _cda_ext_omit
        cmp     al,':'          ; ':'があったら拡張子指定はなかった.
        jz      _cda_ext_omit
        cmp     al,'.'          ; 拡張子指定あり.
        jz      _cda_ext_exist
        loop    _cda_ext_lp
;
_cda_ext_omit:
        pop     di
        jmp     _cda_set2
;
_cda_ext_exist:
        pop     di
_cda_ret:
        pop     di
        pop     cx
        pop     ax
        ret
;
_cda_set0:
        mov     byte ptr[di],'Y'
        inc     di
_cda_set1:
        mov     byte ptr[di],'*'
        inc     di
_cda_set2:
        mov     byte ptr[di],'.'
        inc     di
        mov     byte ptr[di],'*'
        inc     di
        mov     byte ptr[di],0
        jmp     _cda_ret
check_omission  endp
;
; ------------------ ダブルワード数値→文字列変換
;
; 入力レジスタ  AXBX=被除数,CX=基数,DS:DI=格納アドレス
; 出力レジスタ  AXBX=商,DS:DI=格納アドレス-1
;
num_str_sub     proc
        push    dx
;
        mov     dx,0
        div     cx
        xchg    ax,bx
        div     cx
        xchg    ax,bx
        add     dl,'0'
        cmp     dl,'9'          ; 一応十六進変換もできるように.
        jbe     _ns0
        add     dl,7
_ns0:
        mov     [di],dl
        dec     di
;
        pop     dx
        ret
num_str_sub     endp
;
; ------------------ カンマ付き十進文字列変換
;
; 入力レジスタ  AXBX=元データ,DS:DI=格納アドレス
; 出力レジスタ  DS:DI=格納アドレス-1
; 破壊レジスタ  AX,BX(=0)
;
num_to_str      proc
        push    cx
        push    dx
        push    si
        mov     cx,10
        mov     si,-1           ; カンマ区切りのカウンタ
_ns_lp:
        inc     si
        cmp     si,3            ; 3桁変換したら
        jnz     _ns_normal
        mov     byte ptr[di],','
        dec     di              ; カンマを挿入して
        mov     si,0            ; カウンタをリセット
_ns_normal:
        call    num_str_sub     ; 文字に変換して
        mov     dx,ax           ; 商が0になるまで
        or      dx,bx
        jnz     _ns_lp          ; 繰り返す.
        pop     si
        pop     dx
        pop     cx
        ret
num_to_str      endp
;
; ------------------ 二桁変換（ゼロパディング）
;
; 入力レジスタ  AXBX=元データ,DS:DI=格納アドレス
; 出力レジスタ  DS:DI=格納アドレス-1
; 破壊レジスタ  AX,BX(最後の商)
;
num_to_str2     proc
        push    cx
        mov     cx,10
        call    num_str_sub
        call    num_str_sub
        pop     cx
        ret
num_to_str2     endp
;
; ------------------ 四桁変換（ゼロパディング）
;
; 入力レジスタ  AXBX=元データ,DS:DI=格納アドレス
; 出力レジスタ  DS:DI=格納アドレス-1
; 破壊レジスタ  AX,BX(最後の商)
;
num_to_str4     proc
        call    num_to_str2
        jmp     num_to_str2
num_to_str4     endp
;
; ------------------ 表示バッファをスペースでうめる.
;
; 入力レジスタ  なし
; 出力レジスタ  なし
;
clear_buf       proc
        push    ax
        push    cx
        push    di
;
        mov     di,offset fnbuf
        mov     cx,bufend - fnbuf
        mov     al,' '
rep     stosb
;
        pop     di
        pop     cx
        pop     ax
        ret
clear_buf       endp
;
; ------------------ 表示バッファを出力する.
;
; 入力レジスタ  なし
; 出力レジスタ  なし
;
flush_buf       proc
        push    ax
        push    bx
        push    cx
        push    dx
;
        mov     dx,offset fnbuf
        mov     bx,H_STDOUT
        mov     cx,bufend - fnbuf
        mov     ah,DOS_HWRITE
        int     21h
;
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
flush_buf       endp
;
; ------------------ ファイル属性の表示
;
; 入力レジスタ  なし
; 出力レジスタ  なし
;
attr_data       db      'a'     ; アーカイブ属性
                dw      color_yellow
                db      'd'     ; サブディレクトリ
                dw      color_green
                db      'v'     ; ボリュームラベル
                dw      color_cyan
                db      's'     ; システムファイル
                dw      color_red
                db      'h'     ; 隠しファイル
                dw      color_blue
                db      'r'     ; リードオンリーファイル
                dw      color_magenta

disp_attr       proc
        push    ax
        push    bx
        push    cx
;
        shl     ah,1
        shl     ah,1
        mov     cx,6            ; 6ビット(0～5)検査する
        mov     bx,offset attr_data
_chk_attr:
        mov     al,'_'
        shl     ah,1
        jnc     _chka_next      ; ビットが0なら'_'を表示し
        mov     al,cs:[bx]      ; さもなければ該当するマークをとり
        call    cs:[bx + 1]     ; 表示色を変更する.
_chka_next:
        call    putchar
        add     bx,3            ; テーブルのつぎのデータを指し
        call    color_normal    ; 表示色はノーマルに
        loop    _chk_attr
;
        pop     cx
        pop     bx
        pop     ax
        ret
disp_attr       endp
;
; ------------------ 発見したファイルの内容を表示する.
;
; 入力レジスタ  なし
; 出力レジスタ  なし
;
disp_dir        proc
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
;
```

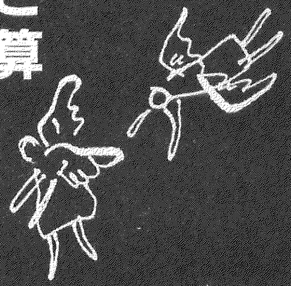

ワンラインNEWS
富士通はマクドネル・ダグラス社とCAD/CAM分野での販売・開発に関して提携をした. これにともない, 富士通はMD社「Parasolid」のライセンスを受けた.

```
        call    clear_buf       ; 表示バッファをスペースでうめておく.
        mov     si,offset dta_name
        mov     di,offset fnbuf
        cmp     byte ptr[si],'.'
        jnz     _setfn          ; 先頭が'.'ならサブディレクトリ.
;
_setdn:                         ; サブディレクトリ名を格納.
        lodsb
        or      al,al
        jz      _setdate
        stosb
        jmp     _setdn
;
_setfn:                         ; ファイル名を格納.
        lodsb
        or      al,al
        jz      _setdate
        cmp     al,'.'
        jz      _setfne
        stosb
        jmp     _setfn
_setfne:
        mov     di,offset extbuf
_setext:                        ; 拡張子を格納.
        lodsb
        or      al,al
        jz      _setdate
        stosb
        jmp     _setext
;
_setdate:                       ; 日付を格納.
        mov     si,[dta_date]
        mov     di,offset datebufe - 1
        mov     ax,0
;
        mov     bx,si
        and     bx,00011111b    ; '日'を処理.
        call    num_to_str2
        mov     byte ptr[di],'-'
        dec     di
;
        mov     bx,si
        mov     cl,5
        shr     bx,cl
        and     bx,00001111b    ; '月'を処理.
        call    num_to_str2
        mov     byte ptr[di],'-'
        dec     di
;
        mov     bx,si           ; '年'を処理.
        mov     cl,9
        shr     bx,cl
        add     bx,1980
        call    num_to_str4
_settime:                       ; 時刻を格納.
        mov     si,dta_time
        mov     di,offset timebufe - 1
        mov     bx,0
;
        mov     bx,si
        and     bx,00011111b    ; '秒'を処理.
        shl     bx,1            ; 秒のデータは1/2で記録されている.
        call    num_to_str2
        mov     byte ptr[di],':'
        dec     di
;
        mov     bx,si
        mov     cl,5
        shr     bx,cl
        and     bx,00111111b    ; '分'を処理.
        call    num_to_str2
        mov     byte ptr[di],':'
        dec     di
;
        mov     bx,si           ; '時'を処理.
        mov     cl,11
        shr     bx,cl
        call    num_to_str2
_setsize:                       ; ファイルサイズを格納.
        mov     bx,dta_sizel
        mov     ax,dta_sizeh
        mov     di,offset sizebufe - 1
        call    num_to_str
;
_dspall:
        call    flush_buf       ; 実際の表示処理.
;
        mov     ah,dta_attr
        call    disp_attr       ; ファイル属性の表示.
        PRINTM  H_STDOUT,<0dh,0ah>
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
disp_dir        endp
;
; ------------------ ファイルサイズ合計の更新処理.
;
; 入力レジスタ  なし
; 出力レジスタ  なし
;
add_total       proc
        push    ax
        mov     ax,dta_sizel
        add     word ptr[size_total],ax
        mov     ax,dta_sizeh
        adc     word ptr[size_total + 2],ax
        pop     ax
        ret
add_total       endp
;
; ------------------ 最初のファイルの検索
;
; 入力レジスタ  DS:DX=ファイル名先頭アドレス
; 出力レジスタ  AX=エラーコード
;
find_first      proc
        push    bx
        push    cx
;
        call    check_omission  ; 省略表記の解決.
        mov     cx,ATTR_ALL
        mov     ah,DOS_FINDFIRST
        int     21h
;
        pop     cx
        pop     bx
        ret
find_first      endp
;
; ------------------ つぎのファイルの検索
;
; 入力レジスタ  なし
; 出力レジスタ  AX=エラーコード
;
find_next       proc
        mov     ah,DOS_FINDNEXT
        int     21h
        ret
find_next       endp
;
; --------------------------------------
;
main    proc
        call    ls_initialize
;
        mov     dx,offset param ; 指定されたファイル名で
        call    find_first      ; 最初のファイルを検索して
        jc      ls_end_n        ; なければエラーコードを返して終わり.
ls_loop:
        call    add_total       ; すでに各種データは取得ずみなので、
        call    disp_dir        ; それを表示し、
        inc     [file_count]    ; ファイル数を一つ増やして
        call    find_next       ; つぎのファイルを検索し、
        jc      ls_total        ; なくなったら終わり.
        jmp     ls_loop
;
ls_total:
        cmp     [file_count],2  ; ファイル数が一個の場合は
        jb      ls_end          ; 合計は表示しない.
;
        call    clear_buf
        mov     bx,word ptr[size_total]
        mov     ax,word ptr[size_total + 2]
        mov     di,offset sizebufe - 1
        call    num_to_str
        call    flush_buf
        PRINTM  H_STDOUT,<0dh,0ah>
;
ls_end:
        mov     al,NORMAL_END   ; 終了コードを返して終わり.
ls_end_n:
        mov     ah,DOS_TERMINATE
        int     21h
;
main    endp

_TEXT   ENDS

_STACK  SEGMENT STACK
;
        dw      256 dup (?)
;
_STACK  ENDS

_DATA   SEGMENT WORD PUBLIC 'DATA'
;
paramc          dw      (?)
param           db      128 dup (?)
;
dta             db      21  dup (?)
dta_attr        db      (?)
dta_time        dw      (?)
dta_date        dw      (?)
dta_sizel       dw      (?)
dta_sizeh       dw      (?)
dta_name        db      13  dup (?)
;
fnbuf           db      8   dup (?)
                db      1   dup (?)
extbuf          db      3   dup (?)
                db      3   dup (?)
datebuf         db      10  dup (?)
datebufe        db      2   dup (?)
timebuf         db      8   dup (?)
timebufe        db      2   dup (?)
sizebuf         db      11  dup (?)
sizebufe        db      2   dup (?)
bufend          db      (?)
;
file_count      dw      (?)
size_total      dd      (?)
;
_DATA   ENDS

        END     main
```

Illustration : Masami Nozue

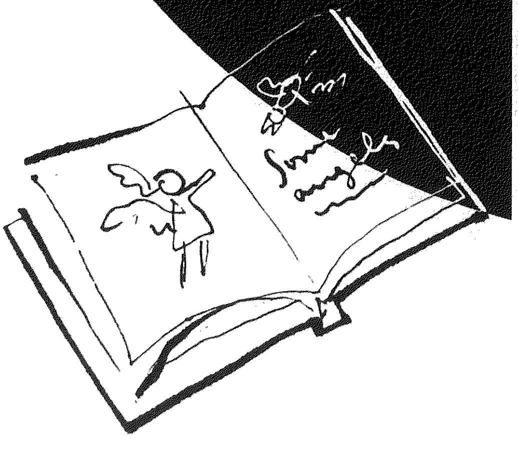
実践C塾 第4回

コードヘルパと浮動小数点演算

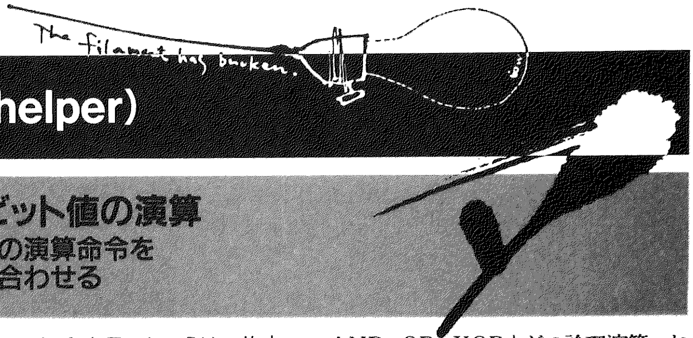


塾頭 水居 浄

今回は，Microsoft CとTurbo Cのコード生成において，ランタイムライブラリと深い関係がある，コードヘルパと浮動小数点演算について述べます．コードヘルパとは，あまり使わない言葉ですが，コンパイラのコード生成のために，自動的に呼び出されるランタイムライブラリです．浮動小数点演算では，80x87を使ったプログラムに対するコンパイラのサービスについて説明しています．そして，両処理系のランタイムライブラリに関する話をはじめます．



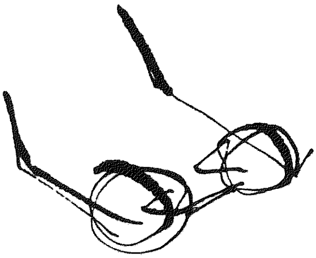
ワンラインNEWS
日本IBMがパソコンの技術情報を公開し，互換機製造支援を呼びかけたことで，三洋電機・三菱電機・シャープなどが製造するAX機の存在が問われている．



1 コードヘルパ (code helper)

コードヘルパとは，あまり聞き慣れない言葉だと思います．これは，コンパイラのコード生成上の手助けのために，自動的に呼び出されるランタイムライブラリのことです．「自動的」というのは，ユーザーがソースプログラム上で明らかに指定しなくても，生成コードではライブラリ呼び出しのコードが展開されてしまうということです．具体的には，Microsoft CやTurbo Cでは次のようなことを行うと，コードヘルパを呼び出すコードが生成されます．

- **32ビット値の演算**
- **構造体の一括操作**



32ビット値の演算
複数の演算命令を組み合わせる

Microsoft CやTurbo Cは，基本的に8086～80286という16ビットCPU上で動作するコードを生成します．現在のバージョンでは，80386以降の32ビットCPUに依存したコードを生成する機能はもっていません．したがって，CPUの演算命令で直接演算できるデータは，16ビット以下のデータになります．すなわち，32ビットデータの演算は複数の演算命令の組み合わせで実現しなくてはならないのです．

Microsoft CやTurbo Cで扱われる32ビットデータとしては，long型やfarポインタ，hugeポインタがあります(float型も32ビットですが，これは別格なので後述します)．long型のデータに対する演算としては，加減乗除などの四則演算，算術シフトやビットごとのAND, OR, XORなどの論理演算，および比較演算，代入演算があります．far／hugeポインタに対する演算は，加減算，比較演算，および代入演算があります．

現実のコード生成としては，たとえば加算などは，図1のような単純な命令の並びですみますから，これは演算を行っている場所に直接コードが展開されます．一方，乗算などはこういう短い命令並びでは処理できませんから，図2のようなサブルーチン呼び出しに展開されます．この呼び出されるサブルーチンがコードヘルパライブラリ（の一部）なのです．

表1～3に，各演算についての両処理系のコード生成方法を示します．

Microsoft Cでは，シフト演算は通

図1　加算の生成コード

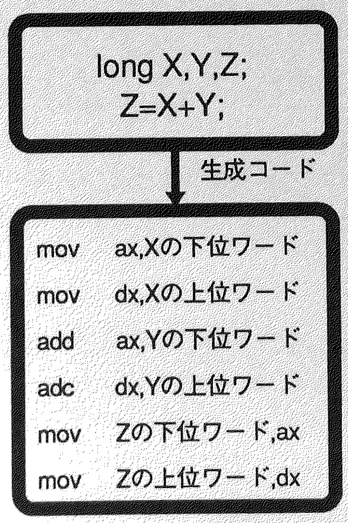
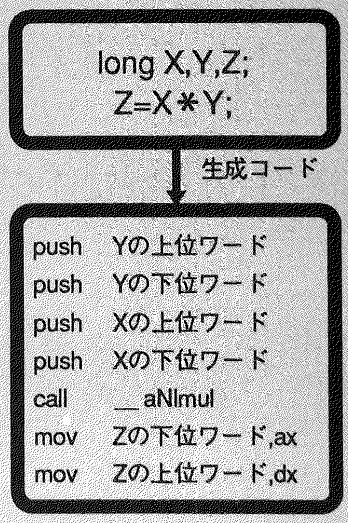


図2　乗算の生成コード



ワンラインNEWS
リコーはパソコンLANシステム「AT&T RICOH STARLAN 10」を東芝の「J-3100」に対応させた．価格は148,000円で年間4,000セットの販売を見込んでいる．

常はインラインコードを生成しますが，サイズを重視する最適化を指定した場合はコードヘルパを呼び出します．

両処理系とも代入演算については，単純な代入演算はインラインコードですが，乗算などのコードヘルパが必要な演算と組み合わせたものは，やはりコードヘルパを呼び出します．

hugeポインタについては，Turbo Cは単純代入以外はすべてコードヘルパ呼び出しになってしまいます．これは前回述べたようにhugeポインタの意味が両処理系で異なり，Turbo Cのほうが複雑な処理を必要としているためです．

これらの32ビット演算コードヘルパについては，Turbo C 2.0とTurbo C++でも少し仕様の違いがあります．Turbo C 2.0では基本的にどのメモリモデルでも，これらのコードヘルパライブラリの呼び出しは，常にfarコールになっていました．しかし，Turbo C++では，メモリモデルやデータのnear／farに応じてnearコール版とfarコール版の2種類が用意されており，適宜使い分けるようです．つまり，そのぶん性能の向上が期待できるのですが，バイナリレベルでの互換性は少し失われたことになります（実際には，Turbo C++のライブラリにはTurbo C 2.0互換のコードヘルパも入っていますから，通常はバイナリでそのままリンクしても問題にはならないようです．もちろん，コンパイルしなおしたほうが性能の向上が期待できますが）．

同様のコードヘルパの仕様変更は，Microsoft Cでも3.0→4.0のときに行われています．同様に，3.0互換のコードヘルパもライブラリに入れておくことで，問題を回避しています．

表1　long型の演算のコード生成

	Microsoft C	Turbo C
加算 +	INLINE	INLINE
減算 -	INLINE	INLINE
乗算 *	HELPER	HELPER
除算 /	HELPER	HELPER
剰余 %	HELPER	HELPER
シフト << >>	INLINE/HELPER	HELPER
論理 & | ^ ~	INLINE	INLINE
比較 == != <= >= < >	INLINE	INLINE
増減 ++ --	INLINE	INLINE
代入 = += -= etc.	INLINE/HELPER	INLINE/HELPER

表2　farポインタの演算のコード生成

	Microsoft C	Turbo C
加算 +	INLINE	INLINE
減算 -	INLINE	INLINE
比較 == != <= >= < >	INLINE	INLINE
増減 ++ --	INLINE	INLINE
代入 = += -= etc.	INLINE	INLINE

表3　hugeポインタの演算のコード生成

	Microsoft C	Turbo C
加算 +	INLINE	HELPER
減算 -	INLINE	HELPER
比較 == != <= >= < >	INLINE	HELPER
増減 ++ --	INLINE	HELPER
代入 = += -= etc.	INLINE	INLINE/HELPER

INLINE：直接演算するコード（インラインコード）を生成する．
HELPER：コードヘルパ呼出しのコードを生成する．

構造体の一括操作
常にコードヘルパが呼び出される

ANSI規格として，型の同じ構造体どうしは一括して代入する演算が許されています．また，構造体を関数の引数として渡したり，関数の戻り値として返したりすることも可能です．構造体は一般に数バイトから数Kバイトの比較的大きなデータになりますから，一括して操作する場合のコード生成は，通常のint型などのデータ型のように単純にはできません．実際には，両処理系で表4のように扱われています．

つまり，Turbo Cでは構造体の一括操作では，常にコードヘルパライブラリが呼び出されます．一方，Microsoft Cではすべてインライン展開します．

以上のように，どういう局面でコードヘルパを呼び出すか，という点で，Microsoft CとTurbo Cは大きく異なります．もちろん，コードヘルパライブラリの仕様（関数名など）も異なりますから，結局，32ビット演算や構造体の一括処理を行ったコードは，両処理系間では（バイ

ワンラインNEWS
富士オフィス機器はパソコン用名刺管理ソフト「ビジュアル名刺管理」を発売した．同ソフトはデータベースソフト「MRDB」と組み合せて富士通の「FM R」で動作する．

ナリレベルでは)，まったく互換性がないということになります．

表4　構造体の一括操作のコード生成

	Microsoft C	Turbo C
代入	INLINE	HELPER
引数として渡す	INLINE	HELPER
戻り値として返す	INLINE	HELPER

2 浮動小数点演算 (floating-point math)

8086系CPUは(80486を除いて)，浮動小数点演算を直接実行する命令をもっていません．したがって，浮動小数点演算は，数値演算プロセッサ80x87を使って処理するか，もしくはライブラリルーチンで処理することになります．

数値演算プロセッサ80x87は，IEEE規格の浮動小数点演算を行う命令をもち，8086系CPUと結合して動作するコプロセッサです．最大80ビットの浮動小数点演算を行うことが可能ですから精度も優れていますし，また，ライブラリルーチンで演算するよりは数十倍高速に演算できます．したがって，8086系CPU上で動作するCコンパイラは，この80x87を使って浮動小数点演算をする前提で設計されています．システムに80x87がないときは，ソフトウェアでエミュレーションするという方式を採用しているものがほとんどです．Microsoft C，Turbo Cもこの方式を採用しています．

ソフトウェアによる80x87エミュレータは，標準のランタイムライブラリの中にあります．また，80x87を使って演算する場合も，エラーハンドリングその他のためにライブラリルーチンが使用されます．したがって，浮動小数点演算を行ったプログラムは，それぞれの処理系のランタイムライブラリに強く依存するものになります．当然ながら，Microsoft CとTurbo Cではバイナリレベルでの互換性はありません．

さらに，80x87を使用するプログラムは一般に機種依存します．これはどうしてかというと，80x87のエラー割り込みのハンドリングが，ハードウェアによっていろいろと異なるためです．80x87は，一般にオーバーフローなどの演算エラーが発生したとき，CPU側に割り込みをかけてエラーの発生を通知します．ところが，この割り込みの入り方がハードウェアの設計によってまちまちになってしまっています．

PC-9801などの一般的な8086/V30マシンでは，8087のエラー割り込みは通常のI/O割り込みと同様に，8259割り込みコントローラを介してCPUに通知します．しかし，IBM PC系の8086マシンでは，NMI割り込みを介してCPUに通知します．一方，80286以降のCPUでは，80x87の割り込み専用の入力端子を備えていますから，80286以降のCPUを搭載したマシンでは一般にその専用割り込み端子を使ってCPUに通知しています．しかし，IBM PC系の80286マシンでは，従来機種との互換性のためにこの機能は使わず，やはりNMI割り込みを使用しています．

こういった違いがあるため，80x87を使用するプログラムでは，エラーハンドラのルーチンをハードウェアに合わせてカスタマイズする必要があります．Microsoft CもTurbo Cも，基本的にはその環境が動作するマシンに合わせたライブラリを提供しています．たとえば，PC-9801用のMicrosoft Cで作成した浮動小数点演算を使うプログラムは，IBM PC上では正しくエラーハンドリングができません．この問題に対処するために，Microsoft Cではエラーハンドラの機種依存する部分だけをソースで提供し，必要ならカスタマイズできるようにしています．しかし，Turbo Cの場合，残念ながらこういうものは提供していません．

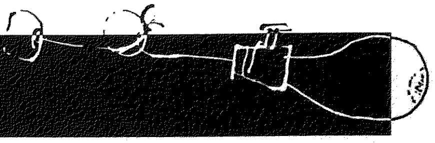
3 ライブラリ (library)

以上，両コンパイラのコード生成上の特徴と違いについて述べてきましたが，両処理系に標準で提供されているランタイムライブラリについても，やはり設計思想の違いから生じる仕様の違いが多く見られます．これから，おもなライブラリについて仕様の違いを検討してみたいと思います．

両処理系とも，ANSI規格への準拠をうたっていますし，実際，ANSI規格で定義されているライブラリの範囲内では，高い互換性をもっています．しかし，ANSI規格のライブラリは，さまざまな環境で共通に実現可能な最低限度のものしかありませんから，実際にMS-DOS上で動作するアプリケーションを開発する場合には，機能的に不足する点も多くあります．そこで，両処理系とも，

ANSIライブラリに加えて，独自仕様のライブラリを追加しています．

Turbo Cの発表当時は，すでにMicrosoft Cが業界標準として定着していたため，Turbo C 1.0のライブラリはMicrosoft C 3.0のライブラリとほぼ互換をとっています．もっとも，Turbo C 1.0の発表当時はすでにMicrosoft C 4.0が発売されていたのですが，4.0で追加されたライブラリとの互換性はあまり配慮されていません．そして，その後，両処理系はバージョンアップのたびにライブラリを拡充してきましたが，これらの追加された関数の仕様もそれぞれ異なり，ほとんど同じような機能を提供していながら，互換性がないという状況になってきています．



4 インクルードファイル (include file)

両処理系の標準のインクルードファイルを表5に示します（OS/2専用のものや，C++専用のものは除いてあります）．

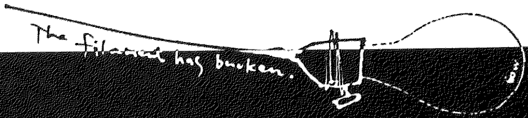
このように，ANSI規格で規定されているインクルードファイルは両処理系ともきちんとサポートしていますが，その他の拡張部分については，それぞれ名前がまちまちになっています．たとえば，chdir () などのディレクトリ操作関数はUNIX互換のものですが，ANSI規格の範囲外であるため，そのプロトタイプ宣言を収めたインクルードファイルは，Microsoft Cではdirect.h，Turbo Cではdir.hと異なってしまっています．

また，malloc () はANSI規格ではstdlib.hでプロトタイプ宣言するのが正しいのですが，他の拡張したメモリ割り当て関数と一緒に宣言できたほうが便利であるために，Microsoft Cではmalloc.h，Turbo Cではalloc.hというインクルードファイルにもプロトタイプ宣言があります．したがって，malloc()を使う場合はstdlib.hの代わりにmalloc.hやalloc.hをインクルードしてもよいのですが，やはりなるべくstdlib.hを使うようにして，少しでも移植性に配慮するべきでしょう．もっとも，_fmalloc () やfarmalloc () などの移植性のない関数を使う場合はどうしようもないのですが…….

表5　処理系とインクルードファイルの関係

	ANSI	MSC6.0	MSC5.1	TC++	TC2.0
alloc.h		－	－	○	○
assert.h	○	○	○	○	○
bios.h		○	－	－	－
bios98.h		－	－	○	○
conio.h		○	○	○	○
ctype.h	○	○	○	○	○
dir.h		－	－	○	○
direct.h		○	○	－	－
dos.h		○	○	○	○
errno.h	○	○	○	○	○
fcntl.h		○	○	○	○
float.h	○	○	○	○	○
graph.h		○	○	－	－
graphics.h		－	－	○	○
io.h		○	○	○	○
jctype.h		○	○	○	○
jstring.h		○	○	○	○
limits.h	○	○	○	○	○
locale.h	○	○	－	○	－
malloc.h		○	○	－	－
math.h	○	○	○	○	○
mem.h		－	－	○	○
memory.h		○	○	－	－
music.h		－	－	○	○
pc98.h		－	－	○	－
pgchart.h		○	－	－	－
process.h		○	○	○	○
search.h		○	○	－	－
setjmp.h	○	○	○	○	○
share.h		○	○	○	○
signal.h	○	○	○	○	○
stdarg.h	○	○	○	○	○
stddef.h	○	○	○	○	○
stdio.h	○	○	○	○	○
stdlib.h	○	○	○	○	○
string.h	○	○	○	○	○
sys¥locking.h		○	○	－	－
sys¥stat.h		○	○	○	○
sys¥timeb.h		○	○	○	○
sys¥types.h		○	○	○	○
sys¥utime.h		○	○	－	－
time.h	○	○	○	○	○
values.h		－	－	○	○
varargs.h		○	○	－	－

ワンラインNEWS
日本電気・東芝について日立製作所も半導体の主要製品である1MビットのDRAMを20%減産した．これは4MビットDRAMへの移行と1MビットDRAMの価格維持が狙い．



5 両処理系に固有のライブラリ

結局のところ，Microsoft CもTurbo CもANSI規格をかなりのレベルで満たしていますから，ANSI規格の範囲内のライブラリを使用している限り，高い互換性を維持できます．しかし，現実にMS-DOS上でアプリケーションを作成する場合，ANSI規格の範囲内のライブラリだけでは機能が不足してしまいます．たとえば，ANSI規格はUNIXやMS-DOSのような階層ディレクトリ構造のファイルシステムを必要条件にはしていないため，ディレクトリの操作などはANSI規格ライブラリの範囲内では行うことができないのです．

したがって，高機能なアプリケーションを作成しようとするほど，どうしても処理系依存のライブラリを使わざるをえなくなります．

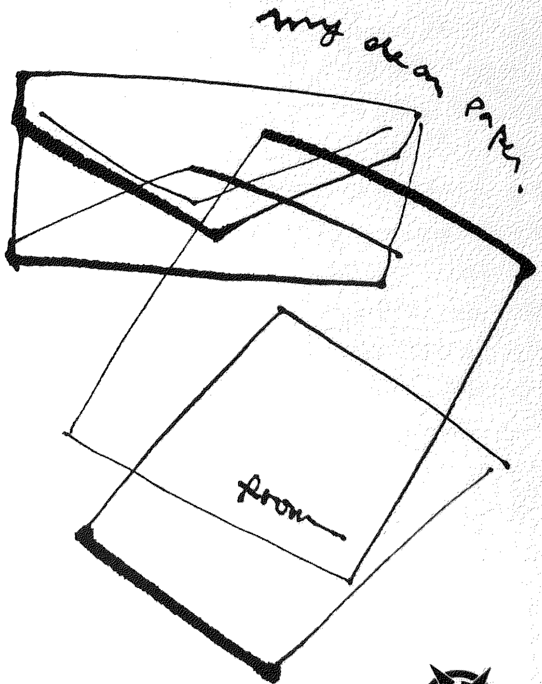
また，ANSI規格に独自の関数を追加する理由はもうひとつあります．それはUNIXやLattice Cなどの「事実上の標準」との互換性のためです．MS-DOSは階層化ディレクトリやI/Oリダイレクトなど，ある程度UNIXもどきの環境をもっています．UNIXのシステムコールや標準ライブラリと互換の関数を適当に用意しておけば，ある程度はUNIX上のプログラムもそのままコンパイルしなおすだけで動くようにすることが可能です．このため，ディレクトリ・ファイル操作ライブラリや，低レベルファイル入出力ライブラリなどは，だいたいUNIX互換の仕様になっています．

Lattice Cについては，この連載の1回目で触れましたが，すでに「業界標準」ではありません．しかし，Lattice CではじめてC言語を知り，その後Microsoft CやTurbo Cに乗り換えてきたというプログラマの人口はかなりのものがあるため，Lattice C互換のライブラリもある程度需要があるのです．こういった「Latticeの遺産」ともいうべきライブラリは，予想以上に根強くパソコンの世界に残っています．

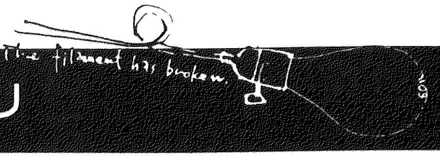
筆者は以前，Macintosh上で動作するLightspeed C(現在はThink Cと改名）というコンパイラのマニュアルを見たとき，Lattice C独特の仕様のライブラリ関数が数多くとり入れられているのを知って，感心した経験があります．

Microsoft CとTurbo Cに標準でサポートされている処理系固有のライブラリは，おもに次のようなものがあります．

メモリバッファ・文字列操作
ディレクトリ・ファイル操作
低レベルファイル入出力
コンソール入出力
メモリ割り当て
プロセス制御
MS-DOSシステムコール
日本語処理
グラフィック処理



6 メモリバッファ・文字列操作ライブラリ



memcpy () やstrcpy () といった，メモリバッファ・文字列操作のライブラリは，おそらくCプログラムでもっとも使用頻度の高いものだと思われます．当然，こういったライブラリはANSI規格に組み込まれているのですが，Microsoft CやTurbo Cにはそれ以外に独自の拡張をしたものを追加しています．表6に各関数名を示します．

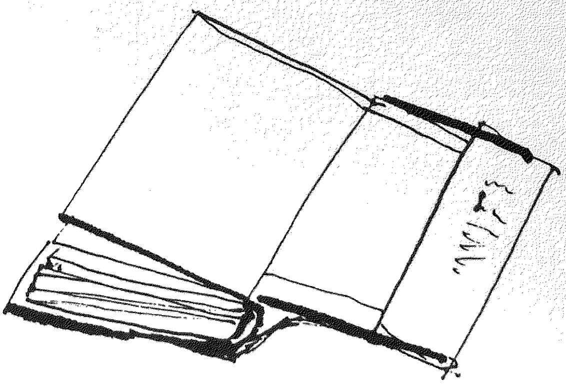
ANSI規格で特徴的なのは，stricmp () やstrlwr () のような文字の大文字・小文字を操作する関数が，すべて外されている点です．これは，ANSI規格のC処理系が，大文字・小文字の区別がないような古いマシン環境でも動作しなくてはならないという理由のためであるようです．もっとも，規格の議論の途中段階ではこれらの関数を取込む構想もあったそうです．stricmp () は，最初strcmpi () という名前でMicrosoft C 3.0が導入しました．

その後，ANSI規格の議論の中で，stricmp () という名前で規格化されることになり，Microsoft C 4.0以降ではstricmp () に名前を変更して，旧バージョンとの互換性を維持するためにstrcmpi () も残す，という処置をとりました．ところが，前述のような理由から，最終的にはstricmp () もANSI規格には残りませんでした．結局，この点ではMicrosoft CのライブラリはANSI規格案に振り回された形になってしまったわけです．

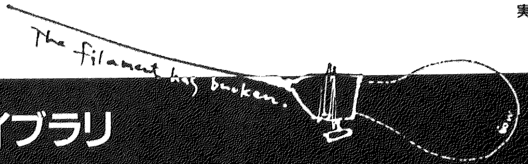
Microsoft C 6.0では，_fmemcpy () や_fstrcpy () などの，メモリモデルに依存しないメモリバッファ・文字列操作ライブラリが追加され，より便利になってきています．

表6　処理系とメモリバッファ・文字列操作ライブラリの関係

	MSC6.0	MSC5.1	TC++	TC2.0	互換性
_fmemcpyなど	○	−	−	−	
_fstrcpyなど	○	−	−	−	
memccpy	○	○	○	○	UNIX
memchr	○	○	○	○	ANSI
memcmp	○	○	○	○	ANSI
memcpy	○	○	○	○	ANSI
memicmp	○	○	○	○	
memmove	○	○	○	○	ANSI
memset	○	○	○	○	ANSI
movedata	○	○	○	○	Lattice C
movmem	−	−	○	○	Lattice C
_nstrdup	○	−	−	−	
setmem	−	−	○	○	Lattice C
stpcpy	−	−	○	○	Lattice C
strcat	○	○	○	○	ANSI
strchr	○	○	○	○	ANSI
strcmp	○	○	○	○	ANSI
strcmpi	○	○	○	○	
strcpy	○	○	○	○	ANSI
strcspn	○	○	○	○	ANSI
strdup	○	○	○	○	UNIX
strerror	○	○	○	○	ANSI
_strerror	○	−	−	−	
stricmp	○	○	○	○	
strlen	○	○	○	○	ANSI
strlwr	○	○	○	○	
strncat	○	○	○	○	ANSI
strncmp	○	○	○	○	ANSI
strncpy	○	○	○	○	ANSI
strnicmp	○	○	○	○	
strnset	○	○	○	○	
strpbrk	○	○	○	○	ANSI
strrchr	○	○	○	○	ANSI
strrev	○	○	○	○	
strset	○	○	○	○	
strspn	○	○	○	○	ANSI
strstr	○	○	○	○	ANSI
strtok	○	○	○	○	ANSI
strupr	○	○	○	○	
swab	○	○	○	○	UNIX



ワンラインNEWS
三菱電機は次世代の光記録技術である光化学ホールバーニングを利用して，1平方センチあたり3千億ビットの情報を記録できる基礎技術を開発した．



7 ディレクトリ・ファイル操作ライブラリ

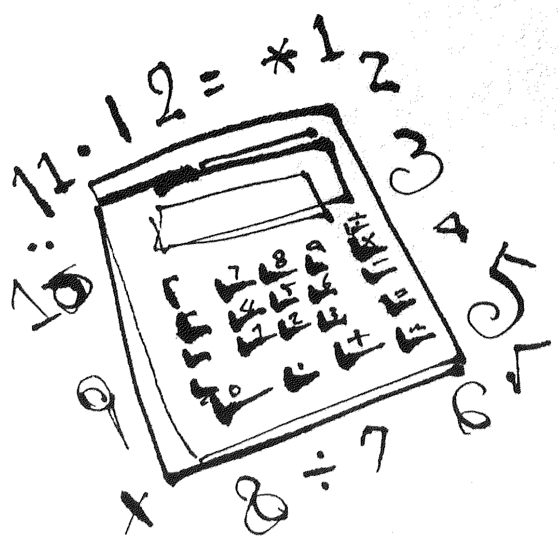
ディレクトリ・ファイル操作ライブラリは，基本的にUNIX互換の仕様のものを採用しています．しかし，UNIXとMS-DOSではファイルシステムに細かい違いがあります．MS-DOSにはUNIXにはない「ドライブ」という概念があり，また「カレントディレクトリ」の意味も少し違います．このため，Microsoft C もTurbo Cも独自に拡張した関数を追加しています．表7に各関数名を示します．

前述のように，ANSI規格はUNIXのようなファイルシステムを必要条件とはしていないため，ディレクトリ・ファイル操作ライブラリは定義されていません．したがって，両処理系ともこのライブラリではUNIXの仕様にならっています．

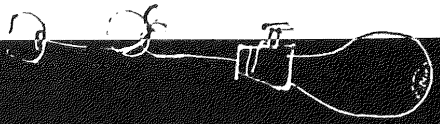
UNIX互換でない拡張関数は，機能的に同じものであっても，ほとんどが両処理系で異なる関数になってしまっています．たとえば，_chdrive()とsetdisk()は，ほぼ同じ機能の関数です（細かい仕様の違いはありますが）．

表7　処理系とディレクトリ・ファイル操作ライブラリの関係

	MSC6.0	MSC5.1	TC++	TC2.0	互換性
access	○	○	○	○	UNIX
chdir	○	○	○	○	UNIX
_chdrive	○	−	−	−	
chmod	○	○	○	○	UNIX
chsize	○	○	○	○	UNIX
filelength	○	○	○	○	
fnmerge	−	−	○	○	
fnsplit	−	−	○	○	
fstat	○	○	○	○	UNIX
_fullpath	○	−	−	−	
getcurdir	−	−	○	○	
getcwd	○	○	○	○	UNIX
_getdcwd	○	−	−	−	
getdisk	−	−	○	○	
_getdrive	○	−	−	−	
isatty	○	○	○	○	UNIX
_makepath	○	−	−	−	
mkdir	○	○	○	○	UNIX
mktemp	○	○	○	○	UNIX
remove	○	○	○	○	ANSI
rename	○	○	○	○	ANSI
rmdir	○	○	○	○	UNIX
_searchenv	○	−	−	−	
searchpath	−	−	○	○	
setdisk	−	−	○	○	
_splitpath	○	−	−	−	
stat	○	○	○	○	UNIX
umask	○	○	−	−	UNIX
unlink	○	○	○	○	UNIX



ワンラインNEWS
ジャストシステムは平成5年からはじまる小中学校のパソコン教育「情報基礎」に対応した「一太郎jump学校パック」の販売を開始した．



8 低レベルファイル入出力ライブラリ

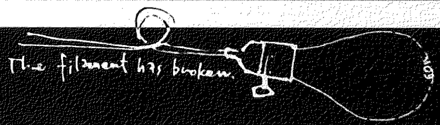
ファイル入出力ライブラリには，ファイルポインタを使って制御するfopen () 系のストリーム入出力ライブラリと，ファイルハンドルを使って制御するopen () 系の低レベルファイル入出力ライブラリがあります．さらに，Microsoft C やTurbo Cには，MS-DOSのシステムコールレベルでファイル入出力を行う，dos_open () 系または_open () 系の入出力ライブラリもあります．

一般にfopen () 系のライブラリのほうが簡単で便利に使用できますし，一方で性能を重視する場合は，システムコールレベルのライブラリを使ったほうが効果は大きくなります．したがって，open() 系のライブラリはMS-DOS上では中途半端な存在であるといえます．むしろ，この系統のライブラリはUNIXとの互換性のために用意されているといった意味あいが強いのです．低レベルファイル入出力ライブラリには，表8のようなものがあります．

これらの関数はANSI規格にはありませんが，UNIXではシステムコールそのものですから，UNIX上のプログラムではよく使われています．このため，UNIX上のプログラムをMS-DOSにもってくる際に有用となります．また，sopenやlock，locking，unlockといった関数は，MS-Networksに対応するための拡張関数です．

表8　処理系と低レベルファイル入出力ライブラリの関係

	MSC6.0	MSC5.1	TC++	TC2.0	互換性
close	○	○	○	○	UNIX
creat	○	○	○	○	UNIX
dup	○	○	○	○	UNIX
dup2	○	○	○	○	UNIX
eof	○	○	○	○	
ioctl	−	−	○	○	UNIX
lock	−	−	○	○	
locking	○	○	○	○	UNIX
lseek	○	○	○	○	UNIX
open	○	○	○	○	UNIX
read	○	○	○	○	UNIX
setmode	○	○	○	○	
sopen	○	○	○	○	
tell	○	○	○	○	UNIX
unlock	−	−	○	○	
write	○	○	○	○	UNIX



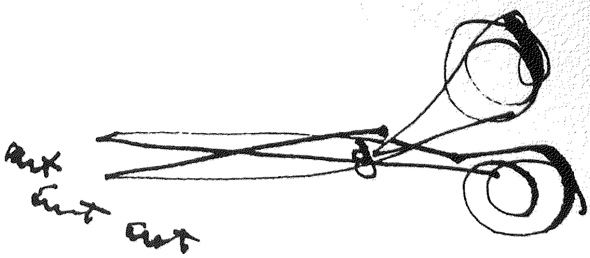
9 コンソール入出力ライブラリ

MS-DOSのいわゆるローナンバーファンクションコール（ファンクション00h～0Ch）を使って入出力を行うライブラリルーチンです．通常のprintf() などでの入出力よりも若干高速に入出力を行うことができます．もともとはLattice Cがサポートしていたもので，Microsoft Cが継承したため，Turbo Cも継承しました．

cgets () とcputs () はMicrosoft CがLattice Cの仕様に追加したものです．

表9　処理系とコンソール入出力ライブラリの関係

	MSC6.0	MSC5.1	TC++	TC2.0	互換性
cgets	○	○	○	○	
cprintf	○	○	○	○	Lattice C
cputs	○	○	○	○	
cscanf	○	○	○	○	Lattice C
getch	○	○	○	○	Lattice C
getche	○	○	○	○	Lattice C
kbhit	○	○	○	○	Lattice C
putch	○	○	○	○	Lattice C
ungetch	○	○	○	○	Lattice C



ワンラインNEWS
富士通は社内のソフトウェア開発担当者の生産性向上のために，支援組織「システム生産技術センター」を設立した．

10 メモリ割り当てライブラリ

malloc () に代表されるメモリ割り当てライブラリは，ANSI規格で規定されています．しかし，前回までに述べてきたようにMS-DOS環境では，8086系CPU特有のnear／farアドレスという概念があるため，実用性を考えるとどうしてもメモリ割り当てライブラリは大幅に拡張せざるを得ません．表10にMicrosoft CとTurbo Cのメモリ割り当てライブラリの各関数名を示します．

このように，Microsoft CとTurbo Cでは拡張部分にはまったく互換性がありません．たとえば，Turbo Cのfarmalloc () は，Microsoft C の_fmalloc () とhalloc () の両方の機能を兼ねています．

Microsoft C 5.1→6.0では，bmalloc () 系のbasedヒープライブラリが追加されています．これは，前回触れた6.0の新機能であるbasedポインタでアクセスできるヒープを管理するライブラリです．また，6.0で追加されたものとしては，_heapminという関数がとくに重要です．

Microsoft Cのメモリ割り当てライブラリでは，一度獲得したメモリ領域は，free () で解放してもMS-DOSには返却しないという仕様になっています（ただし，halloc () だけは例外で，hfree () で解放した領域はMS-DOSに返却されます)．このため，従来は子プロセスを起動するようなプログラムでは，malloc () や_fmalloc () ではなくhalloc () を使うようにしないと，メモリを有効利用することができませんでした．6.0で追加された_heapmin () (および_fheapmin ()) は，この問題を解決するものです．この関数を呼び出すと，それまでにfree () (および_ffree ()) で解放した領域をMS-DOSに返却します．したがって，6.0では子プロセスを起動するようなプログラムでmalloc () や_fmalloc () を使用してもメモリを有効に利用することが可能になりました．

もちろん，Turbo Cのmalloc () やfarmalloc () では，こういう問題は最初から発生しません．free () またはfarfree () で解放した領域はただちにMS-DOSに返却されます．

次回は，ひきつづき両処理系のライブラリについて比較検討します．とくに，MS-DOSシステムコールに関するライブラリは，実用的なアプリケーションを開発する際にはかなり重要なものになりますので，細かく検討したいと考えています．

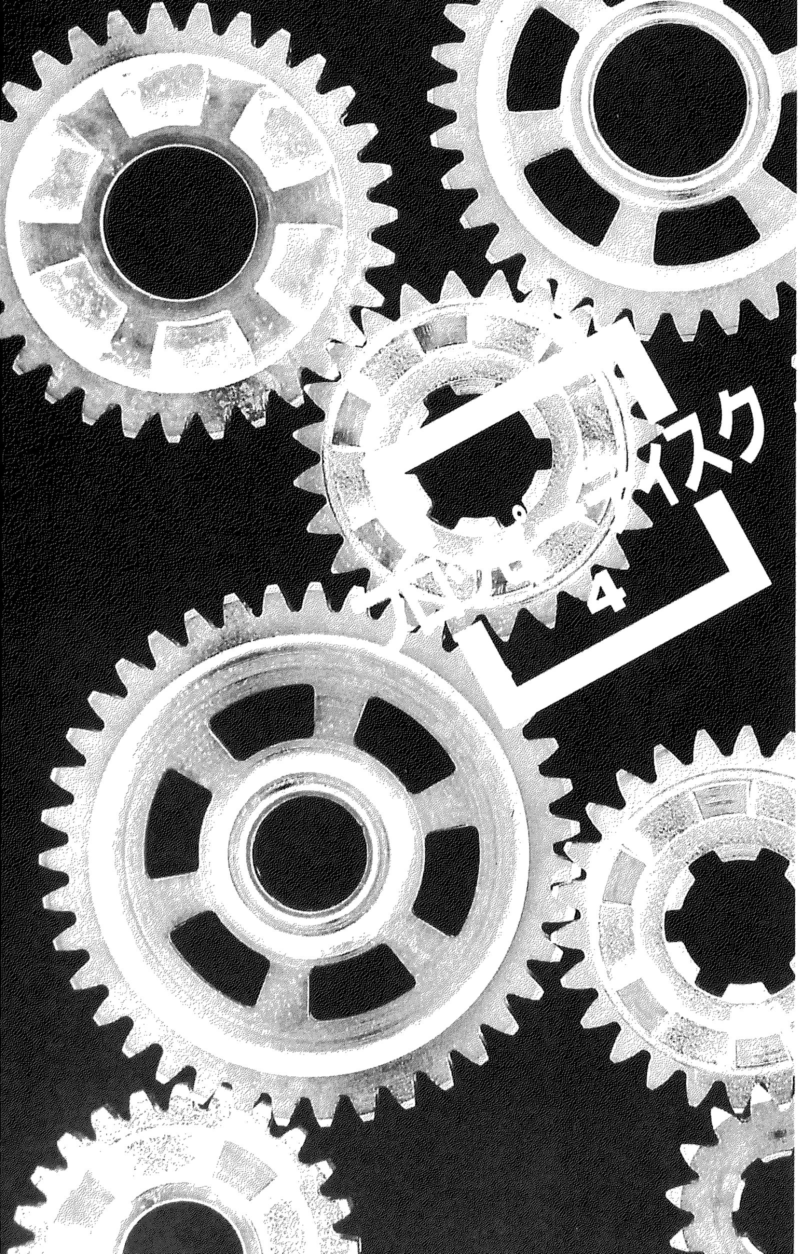
表10　処理系とメモリ割り当てライブラリの関係

	MSC6.0	MSC5.1	TC++	TC2.0	互換性
alloca	○	○	–	–	
_bmallocなど	○	–	–	–	
brk	–	–	○	○	UNIX
calloc	○	○	○	○	ANSI
coreleft	–	–	○	○	
_expand	○	○	–	–	
_fmallocなど	○	○	–	–	
_freect	○	○	–	–	
farmallocなど	–	–	○	○	
free	○	○	○	○	ANSI
halloc	○	○	–	–	
_heapadd	○	–	–	–	
heapcheck	–	–	○	–	
heapcheckfree	–	–	○	–	
heapchecknode	–	–	○	–	
_heapchk	○	○	–	–	
_heapmin	○	–	–	–	
_heapset	○	○	–	–	
heapwalk	–	–	○	–	
_heapwalk	○	○	–	–	
hfree	○	○	–	–	
malloc	○	○	○	○	ANSI
_memavl	○	○	–	–	
_memmax	○	○	–	–	
_msize	○	○	–	–	
_nmallocなど	○	○	–	–	
realloc	○	○	○	○	ANSI
sbrk	–	○	○	○	UNIX
stackavail	○	○	–	–	

Reintroduction to **HARDWARES**

プログラマのためのハードウェア再入門

鎌田正寿



パソコン，ワークステーション，ミニコン，大型コンピュータなどで使われる二次記憶装置にはさまざまなものがあります．フロッピーディスク，ハードディスク，オープンテープ，カセットテープ，CD-ROM，光磁気ドライブ……などなど．ふだんパソコンをおもに使っている方なら，この中でいちばん身近な二次記憶装置はフロッピーディスクでしょう．フロッピーディスク（以下FD）にも，いろいろな大きさやフォーマットが存在し，使うフロッピーの種類の選び方や互換性などについては，ある程度の知識がないとデータの交換はできないことになります．

大きさのいろいろ

現在は，5.25インチ，3.5インチの2つがおもに使われています．

ほかには，次のようなサイズのものがあります．

8インチ

ミニコンなどとのデータ交換用に今も使われている

3～4インチ

ほとんどが3.5インチに淘汰されてしまった

2～3インチ

電子磁気カメラ用など

基本的な動作や用語

FDは，円盤状になっていて円周方向に回転しています．この回転する速さのことを，「回転速度」と呼び，その単位はrpm（1分間あたりの回転数）です．

次に，このFDの半径方向に移動できる「ヘッド」が付いています．ヘッドは，リード・ライトヘッド以外にもイレースヘッドがありますが，ソフト的に見た場合には意識する必要はありません．ヘッドの数は両面用（つまりほとんどの場合）は2つで，それぞれをヘッド0，ヘッド1と呼びます．

ヘッド位置が一定のときには，FDは回転していますので，そのヘッドが通過する場所は円となり，この円のことを「トラック」と呼びます．トラックは外側から順番に，トラック0，トラック1というように呼びます．

同じトラック番号を（ヘッド0，ヘッド1の2つ）集めるとちょうど円筒状の形になり，これをシリンダと呼びます（FDだと多くても2つしかありませんので，円筒という感じはしないかもしれませんね．ハードディスクのように10個くらいのヘッドがある状態を想像してください）．

ここまでに，回転速度，ヘッド，トラック，シリンダの4つが出てきました．場合によっては，シリンダという言葉を使ってないこともあるかもしれません．その場合は，トラックの番号の付け方を片側のヘッド分だけではなくて，次のように番号を付けて呼んでいるはずです．

トラック0：シリンダ0のヘッド0
トラック1：シリンダ0のヘッド1
トラック2：シリンダ1のヘッド0
⋮

1つのトラックには，複数のセクタと呼ばれる単位が存在しています．

なぜかセクタだけは，1から番号が付けられています．1つのセクタにはその種類によって128バイト，256バイト，512バイト，1024バイトのデータを入れることができます．

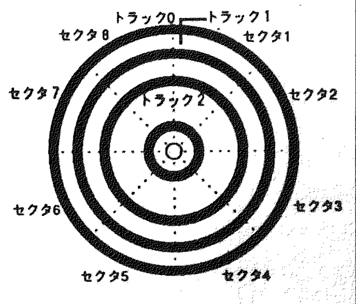
単密度と倍密度

単密度と倍密度の違いはビットごとの記録方式にあります．

単密度で使われる記録方式はFM記録で，倍密度で使われる記録方式はMFM記録です．FDに記録されるデータは，シリアルデータですのでデータをリードする際には，1ビットのデータがどこにあるかを何らかの方法で知る必要があります．

これをどのようにしているかというと，データとともにクロックデータも書き込んでおくのです．実際のFMとMFMの記録方式の違いについてはここでは説明を省きますが，以下の点だけ頭の片隅にでも入れておいてください．

トラックとセクタ



- **単密度＝FM，倍密度＝MFM　で違いは記録方式**
- **他の条件が同じ場合は，MFMはFMに対して2倍の容量となる**

実際には，この単密度は現在はあまり使われていません．8インチなどで1S128というフォーマットが出てくるときがあります．そのときのフォーマットが単密度です．

よく使われている2DDや2HDは，両方とも倍密度でMFMで記録されています．

倍密度と高密度

2DDと2HDの違いで，何が違うかというとデータの転送速度が違います．

2DDに対して，約2倍のスピードでデータをリード・ライトするのが2HDです．記録方式（FM／MFM）は同じです．

データの転送速度が2倍になりますので，記録されるデータの容量も約2倍になります．

データの転送速度は，2DDは250Kビット／秒または300Kビット／秒です．2つの転送速度があるのは，回転数の違いによるもので，本来は300rpmで250Kビット／秒が本当というかオリジナルなのですが，2HDの1.2MBタイプは360rpmであるために，同じ回転速度で2DDもリード・ライトするドライブが存在しています．これらの2つの2DDの転送速度の違いは，ハードウェアが吸収するので，ソフト的には区別する必要はありません(300Kビット／秒のほうが2割速くリード・ライトできるという違いはあります)．

2HDの転送速度は，常に500Kビット／秒になっています．

2HDと2DDの違いは媒体にもあります．よく2DDの媒体を2HDとしてフォーマットして使えるのだろうか，という質問を耳にします．実際はライトした直後であればエラーも出ずに使えてしまえることも多いのですが，時間が立つにつれ信頼性が非常に下がってしまいます．エラーで読めないことがあっても，あきらめがつく場合はそのような使い方もできますが，信頼性が無視できないくらいに下がる，ということだけは憶えておいてください．

もし，そのような使い方をした場合には，できるだけ早く正しい媒体にコピーし直す必要があります．

シリンダ数

もともとの8インチでは，ドライブあたり77シリンダとなっていました．現在，5.25インチや3.5インチでは80シリンダが多く使われています．

ほかには，40シリンダがあり，80シリンダのドライブでこの40シリンダで書き込んだデータを読むには，2シリンダ分を1シリンダとして扱います．このときは，読み込みだけが可能です．40シリンダのドライブで書き込んだデータを80シリンダのドライブで読み込み，同じ80シリンダのドライブで書き込むことはできますが，そのデータを40シリンダのドライブでリードすることは物理的にできません．

セクタあたりのバイト数

単密度(FM)の場合には，128，256，512のいずれかです．

倍密度（MFM）の場合には，256，512，1024のいずれかです．

現在多く使われているセクタあたりのバイト数は，512バイトまたは1024バイトです．LSIによっては，それ以上の2048バイトや4096なども可能ですが，互換性やエラーレートなどの理由により，使われることはほとんどありません．

トラックあたりのセクタ数

フォーマットをしていない状態のトラックあたりの総容量は，転送速度と回転速度と記録方式から決まりますが，これをすべてデータを入れる領域として使用できるわけではありません．このフォーマットをしていない状態からデータをリード・ライトできるように設定を行う操作をフォーマットと呼びます．フォーマットには，規格および事実上の規格が存在し，それによってトラックあたりのセクタ数は決まっています（表1，2，3）．

このほかに，MFM，512バイト／セクタ，8セクタ／トラックのフォーマットも国産のマシンではよく使われます．

3.5インチの2HD，1.2MBと1.44MB

国産のマシンで使われている3.5インチは，5.25インチと同じ約1.2MBの容量があり，IBM PCなどで採用さ

表1　8インチおよび5.25/3.5インチの500Kビット/秒タイプ

	バイト/セクタ	セクタ/トラック
FM	128	26
FM	256	15
FM	512	8
MFM	256	26
MFM	512	15
MFM	1024	8

表2　5.25/3.5インチの250Kビット/秒タイプ

	バイト/セクタ	セクタ/トラック
FM	128	16
FM	256	9
FM	512	5
MFM	256	16
MFM	512	9
MFM	1024	5

表3　3.5インチの500Kビット/秒，300rpmタイプ

	バイト/セクタ	セクタ/トラック
MFM	512	18

ワンラインNEWS
富士通は91年度，半導体メモリ生産拠点の岩手工場に400億円の設備投資をする．4MビットDRAMと，次世代の16Mの量産に備える．

れている3.5インチは約1.44MBの容量があります．これらの違いは，ドライブそのものの違いです．

1.2MBのドライブでは，回転速度は360rpmで，1.44MBのドライブでは300rpmになっています．512バイト／セクタの場合だと，1.2MBタイプではトラックあたり15セクタですが，1.44MBタイプでは回転速度が2割遅い分だけ2割多くセクタを入れることができ，18セクタ／トラックとなっています．

フォーマットの種類

現在使われているフォーマットには，次のような種類があります．フォーマット後のディスク容量の計算は，「ヘッド数」×「シリンダ数」×「セクタ数」×「セクタ当たりのバイト数」です．

フォーマット後の総容量は8セクタ／トラックの場合は，次のようになります．

2DDのフォーマット

ヘッド数	：2（ヘッド／ドライブ）
シリンダ数	：80（シリンダ／ドライブ）
バイト数	：512（バイト／セクタ）
セクタ数	：8または9（セクタ/トラック）
回転速度	：300または360（rpm）
転送速度	：250または300（Kビット／秒）

2×80×8×512＝655360(byte)

＝640（KB）

また，9セクタ／トラックの場合は，次のようになります．

2×80×9×512＝737280(byte)

＝720（KB）

2HD（512バイト／セクタ，1.2MBタイプ）のフォーマット

ヘッド数	：2（ヘッド／ドライブ）
シリンダ数	：80（シリンダ／ドライブ）
バイト数	：512（バイト／セクタ）
セクタ数	：15（セクタ／トラック）
回転速度	：360（rpm）
転送速度	：500（Kビット／秒）

フォーマット後の総容量は

2×80×15×512＝1228800(byte)

＝1200（KB）

となります．このフォーマットは5インチでおもに使われます．2HCと呼ぶ場合もあるようです．

2HD(1024バイト／セクタ，1.2MBタイプ）のフォーマット

ヘッド数	：2（ヘッド／ドライブ）
シリンダ数	：80（シリンダ／ドライブ）
バイト数	：1024（バイト／セクタ）
セクタ数	：8（セクタ／トラック）
回転速度	：360（rpm）
転送速度	：500（Kビット／秒）

フォーマット後の総容量は，次のようになります．これは，5.25インチおよび国産の3.5インチのMS-DOSマシンで多く使われます．

2×80×8×1024＝1310720(byte)

＝1280（KB）

2HD(512バイト／セクタ，1.44MBタイプ）のフォーマット

ヘッド数	：2（ヘッド／ドライブ）
シリンダ数	：80（シリンダ／ドライブ）
バイト数	：512（バイト／セクタ）
セクタ数	：18（セクタ／トラック）
回転速度	：300（rpm）
転送速度	：500（Kビット／秒）

フォーマット後の総容量は，次のようになります．

2×80×18×512＝1474560(byte)

＝1440（KB）

参考資料：
CQ出版　『フロッピ・ディスク装置のすべて』　高橋昇司著
TEAC(FD-235GF-121-021)『マイクロフロッピーディスク仕様書』

COLUMN　セクタスキュー

セクタは，物理的に1，2，3と並んでいるとは限りません．なぜかというとセクタ1をリード・ライトして処理する間にFDは回転を続けていますので，次にセクタ2をリード・ライトしようとするときに，回転待ち状態が起きてしまうことがあるからです．1つ飛びで8セクタ／トラックの場合にはセクタを

1→5→2→6→3→7→4→8

というようにフォーマットします．こうすることによって，回転待ち時間分早くすることが可能になります．

しかし，現在多くのマシンのFDインターフェースで使われているLSIでは，複数のセクタを一度にリード・ライトできる機能をもっていますので，このようなセクタスキューを使うことはめったになくなってしまいました．

98活用アセンブラ技法

ビデオシステム編　第4回

アセンブラと高級言語のリンク・および常駐サブルーチン

星野 操

今回はアセンブラのマクロを利用して，簡単に高級言語からアセンブラで記述された手続きを呼び出す方法と，使い方しだいで便利に活用できる常駐プログラムの基本的な機能である，多重割り込みを利用した常駐のチェック，領域の解放，使用する割り込みベクタの設定，割り込み番号の取得などについてひととおり解説します．

utils.incの修正

_ext_funcの機能とextrn宣言を一度に実現

2回目の連載で掲載したutils. incの一部訂正をします．マクロextfuncの定義ですが，これだけでは関数名のextrn宣言がなされていませんでした．このため，実際にext_funcを使用する場合には，さらにextrn宣言を続ける必要があります．

そこで以前のext_funcを_ext_funcに変更し，またfuncから呼び出すマクロ名も変更することによって以前の機能を確保し，さらに新しいext_funcを定義して，_ext_funcの機能とextrn宣言を一度に実現するようにします．

変更後の部分リストを，リスト1(utils. mod) に示します．

リスト1

```
        .186

_ext_func       macro   nm,arg
        $xlist
        $setn   $val,0                  ;; count number of argument
        irp     x,<arg>
                $inc    $val
        endm
        $setn   $a_&nm,%$val            ;; register it
nm      macro   argl                    ;; function calling macro definition
        $setn   $val,$a_&&nm            ;; get number of declared argument
        irp     x,<argl>
                $list
                push    x
                $xlist
                $dec    $val            ;; and compare it to actual ones
        endm
        if      $val ne 0               ;; argument number differnt,
                .err                    ;; cause error
                $if2
                        $list
                        %out    UTILS Error: argument error on calling nm
                        $xlist
                $endif2
        endif
        $list
        call    _&&nm                   ;; call under-scored function
        $xlist
        endm
        $list
        endm

ext_func        macro   nm,arg
        $xlist
        _ext_func       nm,<arg>
        extrn   _&nm:near               ;; extern definition
        $list
        endm

func    macro   nm,arg,opt
        $xlist
        _ext_func       nm,<arg>        ;; define schema of function
        ifb     <opt>


```


ワンラインNEWS
アシストはパソコン用毛筆ソフト「アシストフォント」を開発した．対応機種は日本電気のPC-9800シリーズで，価格は9,700円．

高級言語とのリンク

さて，前回は外字を利用して枠線を引くサブルーチンを作成し，それを呼び出すアセンブラの「完全な」プログラムを書きました．しかし，このようなサブルーチンが実際に単独のプログラムとして用いられることはまずありません．

連載第1回で述べたように，プログラム全体をアセンブラだけで書くことはあまりなくなってきています．実際には，高級言語から呼び出されるサブルーチンの一部のみをアセンブラで書くことが多いのです．そしてそれさえも，インラインのアセンブラで展開してしまうことが少なくありません．

ですから，この高級言語とのインターフェースのとり方というのは，習得しておく必要があると思います．

拡張proc宣言

以前のアセンブラでは，この連載で使っているような各種簡略記法は許されていませんでした．セグメントはそれぞれ名前をつけて定義する必要があり，パラメータのスタック渡しの方法についても明示的に記述する必要がありました．

そのため，高級言語から呼び出されるサブルーチンをアセンブラで書く場合には，内容が空の関数を書いてそれをコンパイルし，コンパイラからのアセンブラ出力を利用して，その中にアセンブラでプログラムを書くという方法をとったりしました．しかし，このような方法は，プログラムのメンテナンスの面で，あまりよいとはいえません．

しかし最近のアセンブラでは，とくに高級言語とのインターフェースを容易にするための工夫がされているので，このような用途には非常に単純な記述ですむようになっています．具体的には，funcマクロで定義しているような手続きの引数宣言によって，自動的にスタックフレームを生成・解放するようにできるのです．

つまり，今回使用しているマクロを使えば，アセンブラの側ではほとんど何も変更することなしに，その手続きを外部の高級言語から呼び出せることになります．ここでは，実際にdraw_lineを高級言語から呼び出せる関数に仕立て上げてみましょう．

line.asmの変更点

メインルーチンなどを取り除き1つのファイルにする

まず，line.asmのメインルーチン，そのほかの部分を取り除いて，1つのソースファイルに仕立て上げます．具体的には，関数get_addressとdraw_lineのみを含むファイルを作ります．

インクルードファイルは必要ですが，end疑似命令のあとのエントリポイントの指定をしてはいけません．また，.stackセグメントの指定もオミットします．これらの指定は，アセンブ

リスト2-①

```
;---------------------------
; 線を引く
; linesub: draw line with border char
;
; $Log: RCS/linesub.asm $
; revision 1.1 TAN 91/01/26 07:28:29
; Initial revision
;
;
;---------------------------
        dosseg
        .model  small,pascal

        masm51                          ; delete if masm

        include condx.inc
        include utils.inc
        include dosfile.inc

left    equ     20h
right   equ     10h
hdouble equ     8
up      equ     4
down    equ     2
vdouble equ     1

;---------------------------
; VRAMのオフセットアドレスを取得
; get_address: get video ram offset address
; argument:
;       x: column address
;       y: row address
; return:
;       ax: offset

func    get_address     <x, y>
        mov     ax,y                    ; 1line=160 bytes
        mov     dx,160
        mul     dx
        add     ax,x                    ; add column*2
        add     ax,x
        ret
endf

;---------------------------
; 四角を描く
; draw_line: draw (rect-angle) line
; argument:
;       x1, y1: position of one edge of line
;       x2, y2: position of another edge of line
;       double: line flag
; return
;       none

func    draw_line       <x1, y1, x2, y2, double> <si di>
        ; check box/line
        @mc
```

ラ，またはCなどのメインルーチンのほうで定義・使用することにします．

public宣言

そのかわりに，この関数を別のファイルから使用できるようにラベルのパブリック宣言をします．

注意しなければいけないのは，宣言するラベル名には，頭にアンダースコアをつける必要があるということです．ラベル名は，_draw_lineになります．draw_lineというシンボルは，関数を呼び出すマクロ命令になっているからです．もちろん，外部でこの関数を使用する場合も，アンダースコアつきの名前で呼び出す必要があります．

draw_lineの変更

draw_line自身の記述ですが，多少変更しています．先月のバージョンでは，doubleの値が0でシングル，非0でダブルラインを引くようになっていましたが，今月のバージョンでは1でシングル，2でダブル，そして0でラインを消すようになっています．AHにCHでANDをとってからCLとORをとるという点は同じです．ただ，CH/CLの値およびこれを修正するためのBH/BLの値が変更されています．

CX/BXレジスタの内容

たとえば縦線を引く場合，CHの最初の値は縦線種と下線ビットをマスクするような値になっています．またCLは，下線ビットおよび線種ビットに値が設定されています．これによって，上線ビットに触ることなく下線ビットを立てることができます．以降の線では，CHは線種・上下線両方のビットをマスクし，CLで上下線両方のビットを立てるようにします．このため，BHに

リスト2-②

```
                mov     ax,x1
                cmp     ax,x2
                @cond   nz
                mov     bx,y1
                cmp     bx,y2
                @cond   nz
        @if
                ; both not equall, draw box
                draw_line       <x1, y1, x1, y2, double>
                draw_line       <x1, y2, x2, y2, double>
                draw_line       <x2, y2, x2, y1, double>
                draw_line       <x2, y1, x1, y1, double>
                ret
        @endif

        cmp     ax,x2                   ; compare x1 and x2
        @if     nz
                ; horizontal line
                @if     nc
                        xchg    x2,ax   ; x1 must be smaller one
                        mov     x1,ax
                @endif

                cmp     double,0        ; setup mask
                @if     z
                        mov     ch,not right
                        mov     cl,0
                        mov     bh,left
                        mov     bl,0
                @else
                        mov     ch,not (right or hdouble)
                        mov     cl,right
                        mov     bh,left
                        mov     bl,left
                        cmp     double,2
                        @if     z
                                or      cl,hdouble
                        @endif
                @endif

                mov     di,x2           ; column counter set
                sub     di,x1
                inc     di

                mov     si,2
        @else
                ; vertical line
                mov     ax,y1
                cmp     ax,y2
                @if     z
                        ret             ; null line
                @else
                @e_if   nc
                        xchg    y2,ax   ; y1 must be smaller one
                        mov     y1,ax
                @endif

                cmp     double,0        ; setup mask
                @if     z
                        mov     ch,not down
                        mov     cl,0
                        mov     bh,up
                        mov     bl,0
                @else
                        mov     ch,not (down or vdouble)
                        mov     cl,down
                        mov     bh,up
                        mov     bl,up
                        cmp     double,2
                        @if     z
                                or      cl,vdouble
                        @endif
                @endif

                mov     di,y2           ; row counter set
                sub     di,y1
                inc     di

                mov     si,160
        @endif

        get_address     <x1, y1>        ; get vram address
        xchg    di,ax           ; save it
        xchg    dx,ax           ; counter set
        mov     ax,0a000h       ; set vram segment
        mov     es,ax

        @begin
                ; get and decode video code
                mov     ax,es:[di]
                shr     al,1
                rcl     ah,1
```


ワンラインNEWS
三菱電機は厚さ0.5ミリという世界で最も薄いICパッケージを開発した．パソコンのHDDに取って代わるものとして注目を集めている．

リスト 2-③

```
                @mc
                        cmp     al,56h/2        ; gaiji high byte
                        @cond   nz
                @or
                        cmp     ah,40h
                        @cond   nc
                @if
                        mov     ax,56h/2        ; set space code
                @endif

                ; modify it
                and     ah,ch
                or      ah,cl

                ; new mask code set
                test    ch,bh
                @if     nz
                        xor     cx,bx
                @endif

                cmp     dx,2
                @if     z
                        shr     bx,1
                        xor     cx,bx
                @endif

                ; encode and write video code
                test    ah,UP or DOWN or RIGHT or LEFT
                @if     z
                        mov     ax,' '
                @else
                        shr     ah,1
                        rcl     al,1
                @endif

                mov     es:[di],ax
                add     di,si
                dec     dx
        @until  z
        ret
endf

        public  _draw_line

        end
```

は上線ビットを設定し，これとCHとのビットテストをすることによって最初の1回目を判定し，条件が満たされた場合にはCH/CLをBXとXORすることによって修正します．BH/BLは，ともに上線ビットが立った値になっていて，これによって適切な修正がなされることになります．

一方，最後の1キャラクタは，下線ビットに触らないようにしなければいけません．このため，CHの下線ビットはセット（クリアしない）し，CLの下線ビットはクリアしなければいけません．それは，先ほど使用したBXの値を右に1ビットシフトし，上線フラグのビットを下線フラグに移すことによってなされます．

一方，線を消去する場合には，マスクする値はそのままで，CLの値をつねに0に保てばよいのです．ただ，CHの線種フラグはマスクしないようにしてやる必要があります．これらの条件を満たすようなCX/BXの値を，引数であるdoubleの値によって設定するように，前の部分が変更されています．

空白キャラクタの処理

最後に，線を消去する場合には線素がなくなってしまったキャラクタの処理をしなければいけません．このようなキャラクタが生成された場合には，空白コードをAXに設定し，そうでない場合には，以前のバージョンのようにシフト命令で外字JISコードを生成します．そして，それを画面に書き戻せばよいわけです．

呼び出し側の記述

pascal記述子

では，次にこれを呼び出す高級言語のプログラム例を示します．linetest.cがそのプログラムです．このプログラムは，ヘッダーファイルのインクルードのあとに_draw_line関数の宣言があります．アンダースコアがついていることと，pascal修飾子がついていることに注意してください．このプログラムはTurbo Cで試しましたが，MS-Cも同様の修飾子をもっているはずです．最新のバージョンでは，_pascalになっているかもしれません．

リスト 3-①

```
/* linetest : draw_line function check program
 * draw_line関数チェックプログラム
 *
 * $Log: RCS/linetest.c $
 * revision 1.1 TAN 91/01/26 07:31:07
 * Initial revision
 *
 */

#include <stdio.h>
#include <stdlib.h>

pascal _draw_line(int x1, int y1, int x2, int y2, int mode);

void wait()
{
    int i;

    for (i = 0; i < 30000; i++)
        ;
}

main()
{
    int i;

```

リスト 3-②

```
	printf("¥033*");

	for (i = 0; i < 24; i++) {
		_draw_line(0, 0, (i * 80) / 24, i, 1);
		wait();
	}

	for (i = 0; i < 24; i++) {
		_draw_line(0, 0, (i * 80) / 24, i, 2);
		wait();
	}

	for (i = 0; i < 24; i++) {
		_draw_line(0, 0, (i * 80) / 24, i, 0);
		wait();
	}

	printf("¥033*");
}

```

pascal宣言時のシンボル処理

このようにpascal宣言することによって，指定した関数はパスカル形式の呼び出し（引数を左からスタックにプッシュし，それらは呼び出された側でポップアウトする）を使用し，かつ関数名にはアンダースコア添付などの処理がされずに大文字に変換されるということになります．

masmでも/mxなどのスイッチを指定すれば，外部シンボルの大文字小文字の区別ができますが，.model疑似命令でpascalモードが指定されていれば，つねに外部シンボルはすべて大文字に変換されます．これは，pascalの規約で大文字小文字の区別をしないと決まっているからです．アセンブラもこの規約に合わせてあり，Cでもpascal修飾子を使用すれば，その規約通りの命名規則が採用されるのです．

プログラムの内容

プログラムは，左上から右下に向かって，まずシングルラインのボックスを重ね書きし，次にダブルラインで重ね書きし，最後にそれらをクリアしていきます．プログラム中では，この_draw_lineはそのままCの関数と同様に使用されています．このようにするだけで，アセンブラで記述した関数を呼び出せるのですから簡単です．

メモリモデル

もちろん，このときにはCなどの高級言語のプログラムと，アセンブラのプログラムのメモリモデルを同一にしてやる必要があります．前掲のlinesub.asmはスモールモデルと指定がありますから，Cのほうもスモールモデルでコンパイルしなければいけません．

プログラムの作成

高級言語とのリンクは簡単そのもの

プログラムを作成するには，最初にアセンブラファイルをアセンブルし，objファイルを作成してから，

```
tcc  linetest.c  linesub.obj
```

などとしてリンクし，実行ファイルを作成します．本当は，tccの引数でlinesub.asmと指定してやれば自動的にアセンブラを起動してアセンブルしてくれるはずなのですが，そのようにすると「upが未定義である」という趣旨のエラーが出てしまいました．

以上のように，高級言語と直接アセンブラで記述した関数をリンクすることは，非常に簡単な作業であることがわかります．

常駐サブルーチン

メモリにルーチンを常駐させ，割り込みで呼び出す

アセンブラで記述したルーチンを呼び出す方法は，このように直接リンクしてプログラムの一部とし，それを呼び出す方法のほかに，そのルーチンを含む常駐プログラムを作成し，それをメモリ内に常駐させて割り込みなどで当該ルーチンを呼び出すという方法があります．

常駐サブルーチンの意味と制限

DOSのデバイスドライバ，そして極論すればDOS自身も，この意味でのメモリ常駐しているサブルーチンであるということができるでしょう．これらが果たす機能というのは，それを呼び出すプログラムの中に直接記述することもできないわけではありません．DOSの文字出力などが存在していなくても，VRAMに直接書き込むようにプログラムすることはできます．しかし，そのようにすると

①簡単な作業のプログラムでも記述が非常にたいへんになる．
②個々のプログラムサイズがとても大きくなる．
③ハードウェアを直接触る記述をしてしまうと，プログラムのポータビリティが悪くなる．
④プログラムの書き手によって同様な機能をもつ，しかし，微妙に異なるサブルーチンが多く生じ，生

ワンラインNEWS
アップルコンピュータジャパンは芝・虎ノ門・大阪・千葉・名古屋などについで，福岡にも「AppleCenter」を設立する．顧客教育やコンサルティングなどを行う．

産性が悪化する.

というような欠点が生じることになります. ですから, DOSのように,「インターフェース」だけを規定したプラットフォームが用意されているのです. よって, このインターフェース様式にのっとれば,「簡単に, 小さなプログラムサイズで, ハードウェアに依存しない, 標準化された」プログラムを書くことができるようになるのです.

では, およそ便利なものはなんでもメモリに常駐させて呼び出せばよいかというと, メモリサイズの制限からそうもいかなくなってしまいます. このようなメモリ内に常駐させるサブルーチンは, インターフェースがしっかりした, 多くのプログラムで使用されるようなサブルーチンであるべきです.

プラットホームとしての用法

ただ, それほど多くのプログラムから使用されるものでなくとも, ハードウェアの相違を吸収するためのプログラムであるなら, それを本体と分離して常駐させる意味はあります. そのようなルーチンを直接当該プログラムとリンクしてしまうと, 機種ごとに異なったプログラムを用意してやる必要が生じます.

しかし, その機種依存の部分だけをまとめて常駐プログラムとし, それを上位のプログラムから呼び出すようにすれば, ハードウェアがかわっても, 常駐部分を入れかえるだけで, 本体になんら変更をすることなくプログラムを実行することができるわけです.

たとえば, プログラムを高速に実行するためにDOSの画面出力を使用せず, 直接VRAMに書き込みをするということはよくあることです. しかし, VRAM構造はPC-9801とそのほかの機種 (J-3100やIBM PCなど) とでは異なっています. ですから, このような振舞をするプログラムは機種に依存してしまうわけですが, もしも画面操作部分のサブルーチンを分離し, 常駐プログラムにしてしまえば, それを呼び出すプログラムを機種非依存にすることは可能です.

今回のような簡単な「枠を描く」, あるいは「線を引く」というルーチンでも, J-3100用やIBM PC用のそれを用意してやれば, 上位プログラムを共用できる可能性があるわけです. このように, 常駐サブルーチンというのは,「場合によっては」有効に使用できる形態となります.

常駐プログラム

常駐終了ファンクションコールにより実現

「常駐プログラム」というもの自体については, いろいろと書籍や記事などもありますから, ここでは簡単に触れるだけにします.

通常のプログラムは, その実行が終了するとメモリから取り去られ, その空いた領域は再使用されます. しかし,「常駐終了」というファンクションコールを使用すれば, そのプログラムはメモリから取り去られることなく, その使用した領域も再使用されることはありません. 結果として, そこに存在しているプログラムやデータ領域は, 他のプログラムからアクセス (呼び出し, 読み書きなど) することが可能となります. つまり, 常駐終了するプログラムにサブルーチンを入れておけば, あとで他のプログラムからそれを呼び出して使用することができるわけです.

常駐プログラム内のサブルーチンを外部から呼び出す方法としては, 通常は割り込みを使用します. これ以外にも, なんらかの方法でエントリポイントのアドレスを取得し, そこをfar callするという方法があります. また, デバイスドライバとして登録し, ioctlを利用して呼び出すこともできないわけではありません.

割り込みを使用する場合でも, あらかじめ固定した割り込みを使用するか, あるいはint2fの多重割り込みを利用するか, なんらかの方法で割り込み番号を設定・取得することができるようにするなどの方法があります.

さらに, きちんとした常駐プログラムにするときには, 常駐時に自分自身がすでに常駐しているかどうかをチェックして多重常駐を避けなければいけません. さらに, 常駐領域を解放することもできるようにする必要があります. このとき, 割り込みベクタなどを使用していれば, 元の値に復元しないといけません.

今回は, 一応これら常駐プログラムに要求される機能をひととおり用意しておくことにします. 多重割り込みを利用して常駐のチェック・領域の解放・使用する割り込みベクタの設定・割り込み番号の取得などを実行します. さらに, 初期化ルーチンは通常のコードセグメントとは別に, 初期化専用のセグメントを用意することにします.

常駐プログラムの構成は, これら主要な機能を1つのファイルにまとめ, さらに使用する割り込みルーチンを別ファイルとします. これによって, 後者のソースのみの変更で, 任意の常駐プログラムを作成することができるようになります.

常駐プログラム主要部分

では，まず常駐プログラムの主要部分であるresident.asmについて説明しましょう．

頭書きの部分は，他のプログラムと同じです．ただ，設定するユーザー割り込みルーチンのシンボルを宣言しています．割り込みルーチンは，通常の関数宣言を利用できないので，ext_funcの利用を避けて，直接extrn宣言をしています．

次のMY_IDは，多重割り込みint2fで自分自身に向けて発行された割り込みであることを認識するためのキーワードです．適当なワード定数を定義しますが，もちろんプログラムを変更した場合には値を変えないといけません．

多重割り込み処理ルーチンとその変数

その次に多重割り込み処理のルーチンの記述があります．このルーチンは，常駐終了時にもメモリ内にとどまっていないといけないので，通常の.codeセグメント内に記述します．

データセグメントには，ユーザー割り込みの設定している割り込み番号と，設定前の割り込みベクタアドレスを格納する変数を宣言しておきます．ここで，my_int_numが－１のときは，ユーザー割り込みが設定されていないことにします．

さらに，コードセグメント内に多重割り込みチェーン用の元のint2f割り込みルーチンアドレスを格納する変数を用意します．これは，データセグメントにおくことは困難です．コードセグメントにおいておけば，データセグメントレジスタの値を呼び出し時のそれに設定したまま，元の処理ルーチン

リスト 4-①

```
;-----------------------------
; 常駐サブルーチン
; resident: resident subroutine
;
; $Log: RCS/resident.asm $
; revision 1.2 TAN 91/01/27 03:22:47
; slightly modified
;
; revision 1.1 TAN 91/01/26 07:29:06
; Initial revision
;
;
;-----------------------------

        dosseg
        .model  small,pascal
        .stack  1000h

        masm51                          ; delete if masm

        include condx.inc

        include utils.inc
        include dosfile.inc

        extrn   _my_int_func:near

MY_ID   equ     2376h

;-----------------------------
; 多重割り込み処理
; int2f: mulitiplexed interrupt entry
; argument:
;       AX: ID
;       BX: function code
;           0 - check residence
;           1 - release program
;           2 - set interrupt vector
;               DX:vector number, -1 for reset vector
;           3 - get interrupt vector
; return:
;       DX: psp of program (function 0/1)
;           interrupt number (function 2/3)
;       AX: broken

        .data
my_vect dd      ?
my_int_num      dw      -1

        .code
vect2f  dd      ?

_int2f:
        ; first, check interface ID
        cmp     ax,MY_ID
        @if     nz
                jmp     vect2f          ; chain to old interrupt
        @endif

        sti                             ; enable interrupt
        push    ds
        mov     ax,_DATA                ; set data segment value
        mov     ds,ax

        ; branch with function code
        or      bx,bx
        @if     z
                ; check residence, return psp
                mov     dx,$psp         ; $psp is a system variable
        @else
        dec     bx
        @e_if   z
                ; release program
                cmp     my_int_num,-1   ; check if interrupt is already active
                @if     nz
                        push    ds              ; save ds
                        mov     ax,my_int_num   ; resume old vector
                        lds     dx,my_vect      ; load vector value
                        dos     <25h,,>
                        pop     ds
                @endif

                ; release int2f vector
                push    ds
                lds     dx,vect2f
                mov     al,2fh
                dos     <25h,,>
                pop     ds

                ; then free environment
                mov     es,$psp         ; first, get environ seg
                mov     ax,es:[2ch]
```


ワンラインNEWS
1990年の半導体生産額ランキングが米データクエスト社から発表された．上位３社は日本電気など日本企業で占められたが，それぞれシェアは低下している．

に制御を移すことができるからです．

処理ルーチンの宣言

さて，先にも述べたように割り込みルーチンはfuncマクロを使用して定義することはできません．なぜなら，proc疑似命令を使用すると，自動的にスタックフレームが生成され，いろいろと不都合が生じるからです．このため，たんにラベルのみを利用しています．ラベル名は，他の関数などと整合をとるために，アンダースコアを前に追加しています．

処理の内容

最初にAXレジスタの値とキーワードとを比較し，これらが異なっていれば元の処理ルーチンにjumpすることにします．そうでなければ，自分あての処理要求と解釈し，まず割り込みを許可したあとDSレジスタの値を設定します．

次に，BXレジスタによるコマンド要求にしたがって，分岐・処理をします．0は常駐チェックをするために使用し，pspのセグメント値をDXに返します．pspの値は，@startupマクロで$pspという変数に設定されています．

BXが1の場合は，常駐領域解放を意味します．ただし，このプログラムが存在しているメモリブロック自体を解放してはいけません．ここでは，環境領域の解放と使用している割り込みベクタの復元にとどめています．まず，必要なら，ユーザー割り込みベクタの復元を行い，それからint2fのベクタの復元を行います．DSレジスタの値は，ベクタ復元後に元の値に戻す必要があります．

それから環境領域を解放します．環境のセグメントはpspのオフセット2chからのワードに入っています．しか

リスト4-②

```
                mov     bx,ax
                dec     ax
                mov     es,ax
                mov     ax,es:[1]       ; check owner of environ seg
                cmp     ax,$psp
                @if     z
                        ; environment is not released
                        mov     es,bx
                        dos     <49h,,,>        ; free environ
                @endif

                ; return psp to release me
                mov     dx,$psp         ; $psp is a system variable
        @else
        dec     bx
        @e_if   z
                ; install/deinstall interrupt
                cmp     my_int_num,-1   ; check if interrupt is already active
                @if     nz
                        push    ds              ; save ds/dx
                        push    dx
                        mov     ax,my_int_num   ; resume old vector
                        lds     dx,my_vect      ; load vector value
                        dos     <25h,,,>
                        pop     dx              ; resume ds/dx
                        pop     ds
                @endif

                mov     my_int_num,dx   ; save interrupt number
                cmp     dx,-1
                @if     nz
                        ; request to install
                        mov     al,dl           ; set interrupt number
                        dos     <35h,,,>
                        mov     word ptr my_vect,bx     ; save interrupt vector
                        mov     word ptr my_vect+2,es

                        ; then, set new interrupt vector
                        push    ds              ; set code segment value to ds
                        push    cs
                        pop     ds
                        mov     dx,offset _my_int_func
                        dos     <25h,,,>        ; al is still active
                        pop     ds
                @endif
                mov     dx,my_int_num
        @else
        dec     bx
        @e_if   z
                ; return interrupt number
                mov     dx,my_int_num;
        @endif

        pop     ds                      ; resume ds value
        iret

;---------------------------
; イニシャライズルーチン
; initialize routines
; define intialize code segment
;

        .data
end_pos db      0                       ; show the end of data segment

r_option        db      '-r'            ; option string definition
i_option        db      '-i'
dup_message     db      'program is already installed',0dh,0ah,0
not_res_message db      'program not installed',0dh,0ah,0
rel_ok_message  db      'program correctly released',0dh,0ah,0
correct_message db      'program correctly installed',0dh,0ah,0
arg_err_message db      'argument range error',0dh,0ah,0
vect23  dd      ?                       ; for ctrl_c vector

init_seg        segment 'stack'         ; special segment for initialize
        assume  cs:init_seg             ; cs: init_seg

;---------------------------
; 文字列の比較
; strncmp: compare two strings
; arguent:
;       p1, p2: buffer address
;       n: number to compare
; return:
;       ax: result, 0 for equall, nz for not equall

func    strncmp <p1, p2, n> <si di>
        mov     dx,0                    ; dx for flag
        mov     si,p1
        mov     di,p2
        push    ds                      ; setup es
        pop     es
```

ワンラインNEWS
日本ビクターはHDD用として世界最小のステッピングモーターを開発した．厚さ6.1ミリで従来の66%薄型化を実現している．

し，解放する前にそのmcbのオーナーが自分自身であることを確認しています．このあたりのアルゴリズムについては，他の書籍などを参照してください．環境のセグメントが他のmcbの中に吸収されてしまっていることも考えられますが，この場合はメモリ解放のDOSコールが失敗するだけなので，あまり問題はないと思われます．

最後に，BX＝0のときと同様にDXにpspの値を返します．これを利用して，int2fを発効したプログラムは，常駐しているプログラム領域自体を解放することにします．

次に，BX＝2のときには，DXに示される割り込み番号にユーザー割り込みエントリを設定します．まず，すでに割り込みがインストールされている場合には，その割り込みベクタを復元します．それから，DXが－１でない場合には割り込みベクタの値を変数にセーブしたあと，ユーザー割り込みエントリを当該割り込みに設定します．よって，－１を指定すれば，割り込みを解放することができるというわけです．

BX＝3/BX＝2の場合には，ユーザー割り込みの番号をDXに返します．これによって，呼び出しプログラムはどの割り込みを使用したらよいか知ることができます．－１だった場合には，適当な割り込み番号を設定すればよいわけです．

以上のような処理をしたあと，DSレジスタを復元してiretで復帰します．proc疑似命令を使用した場合には，iretではスタックフレームの解放も行われませんから，いろいろと不都合があるのです．

プログラムの非常駐部分

常駐時に必要なのは，この多重割り込み部分だけであり，実際に常駐する

リスト 4-③

```

        @begin
                cmp     n,0
        @while  nz
                lodsb                   ; compare characters
                scasb
                @if     nz
                        mov     dx,1    ; non zero value
                        @break
                @endif

                or      al,al           ; check string tail
                @if     z
                        @break
                @endif
                dec     n               ; decriment counter
        @repeat

        mov     ax,dx                   ; set return value
        ret
endf

;-------------------------
; ASCIIから16進数への変換
; atoh: ascii to hexadecimal
; argument:
;       buf: buffer address
; return:
;       ax: converted value
;

func    atoh    <buf>
        mov     bx,buf
        mov     ax,0
        mov     dh,0

        @begin
                mov     dl,[bx]         ; get character
                inc     bx
                or      dl,dl
        @while  nz
                cmp     dl,'0'
                @if     b
                        @break          ; bad character
                @else
                cmp     dl,'9'
                @e_if   be
                        sub     dl,'0'  ; convert
                @else
                and     dl,not 20h      ; capitalize character
                cmp     dl,'A'
                @e_if   b
                        @break            bad character
                @else
                cmp     dl,'F'
                @e_if   be
                        sub     dl,'A'-10       ; convert
                @else
                        @break
                @endif

                mov     cl,4            ; shift
                shl     ax,cl
                or      ax,dx
        @repeat

        ret
endf

;-------------------------
; 非常駐終了する関数
; teminate: terminate function
;
; argument:
;       status: exit status
;
func    terminate       <status>
        ; restore int23 vector
        lds     dx,vect23
        mov     al,23h
        dos     <25h,,>

        ; terminate
        mov     ax,status
        dos     <4ch,,>
endf

dummy_int:
        iret

```


ワンラインNEWS
リコーは「割り込み機能」を搭載したワープロ「リポートL500シリーズ」「リポート5500シリーズ」を発売した．同機能でオフィスでのワープロの共用が可能になる．

ために使用するルーチンを常駐させる必要はありません．ですから，これらは別のセグメントに格納することが好ましいといえます．

そこで，その非常駐命令を記述するセグメントの前に，それが利用するデータエリアの宣言をしておきます．end_posはデータセグメントの終端を示すラベルであり，それ以降のシンボルは非常駐領域です．このため，residentモジュールは，つねに「プログラムの最後に」リンクする必要があります．

そのあとには，オプション検出のための文字列，またエラーメッセージなど，最初にのみ使用される文字列が定義されています．そして，_c割り込みエントリ23hのベクタをセーブするための変数を用意します．

非常駐セグメント宣言

その次に，非常駐プログラム部分を記述するセグメントを用意します．ここでは.codeなどの簡略セグメントは利用できません．きちんとしたsegment疑似命令を使用する必要があります．今回は，init_segというセグメントを定義しています．クラス名として'stack'を使用していることに注意してください．

dossegで簡略セグメント配置を使用している場合には，普通のセグメントはコード領域とデータ領域の間に配置されます．ですから，非常駐部分として切りはなすことはできなくなってしまいます．クラスstackを指定すると，これはスタック領域にとられるので，プログラムのいちばん最後の部分になります．切りはなすのには都合がよいわけです．おまじないの1つとでも考えてください．

非簡略セグメントを利用した場合に

リスト4-④

```
;-------------------------------
; 初期化部分
; initial: initialize part
;

initial:
        @setup                          ; parse command line

        ; disable ^c interrupt
        mov     al,23h                  ; get ^c vector
        dos     <35h,,>
        mov     word ptr vect23,bx      ; save it
        mov     word ptr vect23+2,es

        push    ds                      ; set dummy interrupt entry
        push    cs
        pop     ds
        mov     dx,offset dummy_int
        dos     <25h,,>
        pop     ds

        mov     ax,MY_ID                ; then check residence
        mov     bx,0
        mov     dx,bx
        int     2fh

        ; duplicate check
        or      dx,dx
        @if     nz                      ; already installed
                ; check R option
                mov     bx,_argv
                strncmp <offset r_option,bx[2],2>       ; compare option
                @if     nz              ; reinstall error
                        mov     dx,offset dup_message
                        @puts <dx>

                        terminate <1>   ; error code
                @endif

                ; release program
                mov     ax,MY_ID
                mov     bx,1
                int     2fh

                ; and release program area
                mov     es,dx
                dos     <49h,,>

                ; message
                mov     dx,offset rel_ok_message
                @puts   <dx>

                ; terminate successfully
                terminate <0>
        @endif

        ; check -r flag
        mov     bx,_argv
        strncmp <offset r_option,bx[2],2>
        @if     z
                mov     dx,offset not_res_message       ; not yet installed
                @puts   <dx>

                terminate <1>           ; error return
        @endif

        ; set my interrupt if need
        mov     bx,_argv                ; check -i option
        strncmp <offset i_option,bx[2],2>
        @if     z
                mov     bx,_argv        ; get argument part
                mov     bx,bx[2]
                add     bx,2
                atoh <bx>

                ; range check
                cmp     ax,100h
                @if     ae
                        mov     dx,offset arg_err_message
                        @puts   <dx>

                        ; terminate
                        terminate <1>
                @endif

                ; set vector
                mov     dx,ax           ; dx: vector number
                mov     bx,2
                mov     ax,MY_ID
                pushf                   ; invoke int2f routine
                call    far ptr _int2f
        @endif
```

リスト4-⑤

```
        ; install vector
        mov     al,2fh                  ; save 2fh vector
        dos     <35h,,,>
        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT
        mov     word ptr vect2f,bx
        mov     word ptr vect2f+2,es
        pop     ds
        assume  ds:_DATA

        push    ds
        mov     dx,_TEXT                ; set 2fh vector
        mov     ds,dx
        mov     dx,offset _int2f
        dos     <25h,,,>
        pop     ds

        ; message output
        mov     dx,offset correct_message
        @puts <dx>

        ; stay resident
        mov     dx,offset end_pos       ; calculate paragraph size
        add     dx,15                   ; data segment length
        mov     cl,4
        shr     dx,cl

        add     dx,_DATA                ; add data segment offset
        sub     dx,$psp

        lds     dx,vect23
        mov     al,23h
        dos     <25h,,,>

        mov     al,0                    ; success flag
        dos     <31h,,,>                ; keep process

init_seg        ends

        end     initial

```

は，CSを必ずassume疑似命令で当該セグメントを指すように指示する必要があります．以降，.data疑似命令は使用しないようにします．

非常駐関数

そのあとに，関数を3つ，それから常駐処理のためのルーチンが記述されています．最初のstrncmpは，指定された長さの文字列が同一であるかどうかを比較するためのルーチンで，オプション比較に利用します．Cのそれとは異なり，大小の比較は省いてしまっています．

プログラムは，lodsbとscasbで1文字ずつ比較をし，ナル文字が現れるか，指定された長さの比較がすむか，比較が一致しなかったかの場合に，ループを抜け出してフラグを返すだけです．SI/DIレジスタを使用しているので，funcマクロに所定の引数をとっています．

次のatohは，アスキー文字列を16進で解釈し，その値をAXレジスタに入れる関数です．16進変換には，DAA命令などを利用した，ひじょうに効率のよい方法があるのですが，ここではていねいに比較・演算をしています．DLレジスタに文字を読み込み，値を補正したあとAXレジスタを4ビット左にシフトし，それにDLレジスタをORすることによって変換を実行します．このatohは，プログラム常駐時にユーザー割り込み番号が指定された場合，それを解釈するために使用します．

teminateは，非常駐終了する関数です．常駐プログラムでは，プログラムのインストール時に割り込みベクタの設定など，いろいろな処理を行います．ですから，中途半端に割り込みベクタが設定された状態のときに，^c入力によってプログラムの実行が中断されると障害がおきます．

このため，最初に^cの中断割り込みint23hのベクタを，何もしないルーチンにすりかえておきます．これは常駐終了時，または非常駐終了時に元に戻さないといけません．いろいろな条件によって，常駐終了しない場合があります．このようなときに，いちいちint23hのベクタを復元するのはめんどうなので，この関数で一括して処理してしまうのです．

非常駐終了時には，プログラムの実行ステータスを返すことができますから，これのみを引数にとるようにします．内容はint23hのベクタを復元したあと，ステータスの内容を読み込み，通常の終了をするだけです．

そのあとに，「すりかえる」int23hルーチンであるdummy_intがあります．これはiretするだけのルーチンです．

初期化部分

最後のイニシャライズパートは，まず@setupマクロで引数の処理などをします．普通のプログラムでは，_argc/_argvをスタックにプッシュしてmainを呼びましたが，ここではそのまま処理を続けています．

まず，先に述べた理由で^cのトラップルーチンをすりかえ，int2fで常駐プログラムの存在チェックをします．ファンクション0を呼び出せば，常駐している場合にはDXにpspのセグメントがセットされます．ですから，DXに0をセットしてint2fを呼び出せば，常駐している場合にはDXが非0，常駐していない場合には0のまま，ということになります．

既常駐時の処理

もし，すでに常駐している場合には，

ワンラインNEWS
日本電気・日立製作所の両社は今春から米国で4MビットDRAMの現地生産を開始する．とくに日立はウェハーからの現地生産はこれがはじめて．

-rオプションによる常駐解除だけを受けつけることにします．そのため，第一引数のポインタとオプション文字列とを比較します．もし等しくなければ，多重常駐のエラーメッセージを出し，terminateを利用して異常終了します．

-rオプションが与えられていれば，int2fのファンクション1によって環境・割り込みベクタの復元を行い，そのあとに得られたpspセグメントによって常駐プログラム領域自体を解放します．それから正常終了のメッセージを出してterminateを呼びます．

未常駐時の処理

さて，常駐していない場合に-rオプションが付加されていれば，未常駐のエラーメッセージを出力して異常終了します．そうでなければ，-iオプションを調べます．もしこれが与えられていれば，そのあとの引数を16進文字列と解釈し，その割り込み番号にユーザー割り込みを設定します．この割り込みを設定するのは，int2fの処理ルーチンを疑似的に割り込み呼び出しすることで行います．フラグをプッシュし，当該エントリをfarコールすれば，割り込みと同様な処理が可能となります．

そして，オプションにかかわらずint2fのルーチンを自前の処理ルーチンにすりかえ，メッセージを出力します．

常駐サイズの計算と常駐処理

常駐終了する場合には，pspセグメントからはじめて，何パラグラフ（16バイト）を残しておくかをDXレジスタに引数として渡す必要があります．これを計算するために，先に定義されたend_posというラベルのオフセットを利用します．これを16で割ったあと，データセグメントとpspのセグメントの差をたすことによって，必要なサイズを計算するわけです．

それから，teminateと同様にint23のルーチンを元に戻し，常駐終了のファンクションコールを実行します．end命令のあとのエントリは，initialとしておきます．

以上のソースによって，常駐プログラムに要求される主要な部分は用意されたことになります．

ユーザー割り込み処理ルーチン

_my_int_funcを書き換えることで機能拡張が可能

次に，residentから要求されるユーザー割り込みルーチンを，lineres.asmに示します．このプログラムは，割り込みで呼び出された部分と，通常の関数として記述されたdraw_lineとのインターフェースを行います．

常駐サブルーチンdraw_lineは，最

リスト5-①

```
;-----------------------------
; ユーザー割り込みルーチン
; lineres: user interrupt routine
;
; $Log: RCS/lineres.asm $
; revision 1.1 TAN 91/01/26 07:28:48
; Initial revision
;
;
;-----------------------------

        dosseg
        .model  small,pascal

        masm51                          ; delete if masm

        include condx.inc
        include utils.inc
        include dosfile.inc

ext_func        draw_line <x1, y1, x2, y2, double>      ; line drawing routine

;-----------------------------
; 割り込みハンドラ
; my_int_func: my interrupt handler
; argument:
;       AH/AL: x1/y1 for draw_line
;       BH/BL: x2/y2 for draw_line
;       DX: double for draw_line
; return:
;       none

_my_int_func:
        sti                             ; enable interrupt

        push    ds                      ; save registers
        push    es                      ; may be broken
        push    dx
        push    cx
        push    bx
        push    ax
        push    si
        push    di

        mov     si,_DATA                ; set data segment
        mov     ds,si

        mov     ch,0
        mov     cl,ah
        mov     ah,0
        mov     si,cx                   ; ax:y1, si:x1

        mov     cl,bh
        mov     bh,0                    ; bx:y2, cx:x2

        draw_line <si, ax, cx, bx, dx>  ; call draw_line

        pop     di                      ; resume registers
        pop     si
        pop     ax
        pop     bx
        pop     cx
        pop     dx
```

ワンラインNEWS
ソニーの米国コンピュータ部門の子会社ソニーマイクロシステムズはラップトップ型WSを現地で販売する．同機は国内で販売されている「NWS-3260」とほぼ同じ．

初にext_funcマクロを利用して宣言されています．そのあとに，my_int_funcの記述があります．割り込みルーチンは，関数としては記述できないので，ここでもたんなるラベルを用いています．割り込みを呼び出す場合には，引数のスタック渡しは通常行われません．そこで，AX/BX/DXレジスタをパラメータとして利用することにします．

ユーザー割り込みルーチンの処理

まず，すべてのレジスタをセーブしたあと，データセグメントを設定します．それから，AH/AL, BH/BLに与えられた引数をワード単位に分解し，それを利用してdraw_lineを呼び出します．最後にレジスタを復元して，iretを実行します．もちろん，このmy_int_funcは最後にpublicと宣言する必要があります．

ローカルスタック

ここで，呼び出される関数はスタックをあまり消費せず，また呼び出しもソフトウェアによって明示的になされることから，割り込み内部でのローカルスタックの設定は行っていません．常駐サブルーチンとして使用する場合には，このような設定をする必要はありないでしょう．

機能の拡張

複数の処理をする必要がある場合には，このユーザー割り込み内部で，レジスタの値によって分岐処理をするようにすればよいことになります．また，値を返すことが必要な場合には，当該レジスタの保存をしなければよいわけです．このように_my_int_funcを適当に書きなおすことによって，常駐サブルーチンの機能を変更することができるようになります．

リスト5-②

```
        pop     es
        pop     ds

        iret

        public  _my_int_func
        end

```

常駐プログラムの作成

リンク時にresidentを最後にもってくることに注意

以上のようにして用意された3つのモジュール，resident・lineres・linesubをそれぞれアセンブル・リンクすれば1つの常駐プログラムができあがります．ここで，先に書いたように，リンク時にはresidentを最後にもってくる必要があるということに注意してください．たとえば，

```
tlink linesub+lineres+resident,resident
```

とすればよいわけです．なお，このときのリンクマップをresident.mapに示します．初期化用のセグメントinit_segが，最後にきていることがわかります．

リスト6

```
 Start  Stop   Length Name               Class

 00000H 0023AH 0023BH _TEXT              CODE
 0023CH 002D9H 0009EH _DATA              DATA
 002E0H 012DFH 01000H STACK              STACK
 012E0H 01542H 00263H INIT_SEG           STACK

  Address         Publics by Name

 0000:0016       _DRAW_LINE
 0000:015C       _MY_INT_FUNC

  Address         Publics by Value

 0000:0016       _DRAW_LINE
 0000:015C       _MY_INT_FUNC

Program entry point at 012E:009C
```

プログラムの実行

こうして作成されたプログラムをresident.exeとすれば，たんにresidentと実行すると常駐し，resident -rと-rオプションを付加すれば，常駐しているプログラムの領域を解放します．また常駐時に-ixxと指定すれば，ユーザー割り込みをあらかじめ特定の割り込みに設定することができるようになっています．

常駐サブルーチンを利用した高級言語プログラム

常駐チェック関数と呼び出し用の関数を追加

最後に，この常駐プログラムを利用する高級言語の使用例を示します．linetst2.cがそれです．このプログラムは，割り込みc0hを常駐ルーチンとのインターフェースに使用します．内容はほとんどlinetestと同じですが，常駐チェック関数と呼び出し用の関数が追加されています．

ワンラインNEWS
日本IBMは半導体のような構造をしたプリント基板を開発した．この基板は表面に導体層と絶縁層を積み上げた構造をしている．

リスト7

```
/* linetst2 : draw_line resident function check program
 * draw_line常駐関数チェックプログラム
 *
 * $Log: RCS/linetst2.c $
 * revision 1.1 TAN 91/01/26 07:31:22
 * Initial revision
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

#define MY_ID 0x2376
#define MY_INTR 0xc0

void wait()
{
    int i;

    for (i = 0; i < 30000; i++)
        ;
}

void check_function()
{
    struct REGPACK reg;

    /* check function */
    reg.r_ax = MY_ID;
    reg.r_bx = 0;
    reg.r_dx = 0;
    intr(0x2f, &reg);

    if (reg.r_dx == 0) {
        fprintf(stderr, "resident function not installed¥n");
        exit(1);
    }

    /* set interrupt number */
    reg.r_ax = MY_ID;
    reg.r_bx = 2;
    reg.r_dx = MY_INTR;
    intr(0x2f, &reg);
}

void _draw_line(int x1, int y1, int x2, int y2, int dbl)
{
    struct REGPACK reg;

    reg.r_ax = x1 << 8 | y1;
    reg.r_bx = x2 << 8 | y2;
    reg.r_dx = dbl;
    intr(MY_INTR, &reg);
}

main()
{
    int i;

    check_function();

    printf("¥033*");

    for (i = 0; i < 24; i++) {
        _draw_line(0, 0, (i * 80) / 24, i, 1);
        wait();
    }

    for (i = 0; i < 24; i++) {
        _draw_line(0, 0, (i * 80) / 24, i, 2);
        wait();
    }

    for (i = 0; i < 24; i++) {
        _draw_line(0, 0, (i * 80) / 24, i, 0);
        wait();
    }

    printf("¥033*");
}
```

常駐サブルーチンの存在チェックとインターフェース割り込みの設定

このような常駐サブルーチンを使用するプログラムでは，必ず最初に常駐していることをチェックしなければいけません．チェックは，アセンブラのプログラムと同様に，キーワードをAXに，BX=DX=0としてint2fを呼び出し，DXの値が変更されているかどうかを利用します．

インストールされていれば，BX=2として，割り込みc0hにユーザー割り込みエントリを設定させます．

常駐サブルーチンの呼び出し

インターフェース用の関数_draw_lineは，AX/BX/DXレジスタに，lineres.asmで指示されているような形式で値を設定し，c0hの割り込みを発効します．これによって常駐しているdraw_lineが呼び出されることになります．

他の高級言語からの呼び出し

dBASEなどから利用することも可能

以上のような呼び出し方をするわけですから，レジスタに値を設定でき，適当な割り込みを発生させられるような高級言語では，このような常駐サブルーチンを呼び出すことができます．たとえば，dBASEのようなデータベースのコマンドファイルの中でこれらを利用することも可能なわけです．

これは，プログラム作成時にリンクすることを必要とする最初の方法よりも，自由度が多少高いといえるでしょう．

PRB
Programmer's Rest Bench

なんと！5年間で1MBあたり約1万5千円から600円以下になった！

パソコンの世界では，「10年大昔」なんていわれるが，何かを調べる機会があって古い雑誌などをめくっていると，つくづくそれを実感することがある．ハードウェアは，加速度的に高機能・低価格になっている．顕著なのがハードディスク．小社刊行の『僕らのパソコン10年史』では，1986年から1987年の98用ハードディスクの価格が次のようなリストになっている．

20MBハードディスクの価格（PC-9800シリーズ用）

1986年	アイシーエム	HD-3020	280,000円
	緑電子	MD-52VA	278,000円
	日本電気	PC-98H51N	298,000円
1987年6月	アイシーエム	HD-20LP	188,000円
	緑電子	DAX-HI	175,000円
1987年12月	アイシーエム	MC-20	125,000円
	緑電子	Little-B	94,000円

1987年の後半に当時としては，低価格のハードディスクが発売されて，本格的な普及期を迎えたことがわかる．

ところが，現在では1987年当時の20MBハードディスクの価格で80～100MBを手にすることができるようになった．さらに，CD-ROMやMOディスクの登場で，一般のパソコンユーザーにもGB，つまりギガバイトというとんでもない大容量記憶装置が身近なものになりつつある．現に編集部にも1GBのMOディスクを買ったという根性の入ったパソコンユーザーがいる．このMOディスク，遅いハードディスク程度のスピードがあるから，りっぱにメインドライブとして使える．価格は，ちょっと高いパソコン程度だという．それにしても，5年前に1MBあたり1万5千円近くしたものが，MOだと6百円程度になってしまうのだから…….

TVゲームとMS-DOSソフトが同時に動く！テラはトリプルCPUマシン

「あれはなんですか．テレビでメガドライブのゲームをやっている横で，Windows 3.0が動いているパソコンは?」というのは，秋葉原から帰ってきた編集部のメンバーの話．店でデモっていた「テラ」のことをいっているのだ．Z80，68000，80286というトリプルCPUのマシンは，どうやら別々のCPUで一度に2つのプログラムを走らせることができるらしい．ジョイスティックとテレビをつないでゲームをして，キーボードとCRTでMS-DOSのソフトを動作せる，そんな使い方ができるのだ．

80286ベースの部分は，IBM PC/AT互換だ．というより，そのモジュールはIBMからOEMされているのだから純正なのだ．VGA搭載で，話題のDOS/Vも当然走る．ATバスのスロットも備えている．68000とZ80の部分はセガのメガドライブ互換．日英両モードのMS-DOSソフトが走りTVゲームもできる．これで，思い切った販売政策がとられれば，おもしろい展開を見せることだろう．

噂が先行していると思っていたら，いつのまにかイベントや店頭に姿を見せていたという，不思議なかたちで登場してきた．春には発売されるらしいから，本誌が店頭に並ぶころには話題沸騰してるか?

HANDY98が当たったぞ！うれしいメッセージがSE-NETに書き込まれた

```
==================================================================
タイトル：ＨＡＮＤＹ９８が当たったぞ！
投 稿 者：Kunitake #34
投 稿 日：平成 03年 02月 01日(金) 午後 01時 20分 32秒
------------------------------------------------------------------
ＰＰ創刊記念プレゼントに応募したらＨＡＮＤＹ９８が当たったぞ！（ワイ！）
今さっき、カミさんからＴｅｌがあり変な物が届いたというので開けてみてもらったら
なんとＨＡＮＤＹ９８だったとは・・・　こいつぁ春から縁起がいいわい！

でも、欲を言えばＮＶなんて言わないからせめてＮＳだったらなぁ・・・
なんてこといったら殴られるか。ハハハ・・
```

というメッセージが書き込まれたのは，小誌オンラインサービス用のBBSであるSE-NETの「ドラゴンポスト」ボードだった．もちろん，応募はがきのなかから公平に抽選をして決めたのだが，SE-NETの登録読者だったとは知らなかった．確認してみたところ，たしかにご本人からのメッセージのようです．おかげで，当編集部がまじめに商品を発送していることが証明されることになった．:-)

ワンラインNEWS
ワイ・イー・データはパソコン用プリンタ分野に進出した．同社はノート型熱転写プリンタを富士通にOEM供給を開始した．

第2種合格！トレーニング

第1回　ハードウェアの基礎Ⅰ

日本工学院専門学校　勝島貴之

過去3年間のハードウェア分野の出題傾向は表1のとおりです．例年，ハードウェアの出題比率は全10問中4～5問というところです．その中で毎年必ずといってよいほど出題される内容として，「コンピュータ内部における情報の表現形式」と「磁気ディスクの容量・処理能力」があります．そこで今回と次回で，これらの分野に関する解説を重点的に行います．

情報の表現形式

基数変換

コンピュータの内部では，“0”と“1”からなる2進数を基本として，データが取り扱われます．私たちが普段使い慣れている10進数の数値表現，‘A’や‘B’という文字表現とは違って，2進数のコードでデータが表現されるわけです．

■ｒ進数表現とｒ進→10進変換

一般にｒを基数(radix)とした数値の表現方法をｒ進数といいます．ｒ進数の各桁は0からｒ-1までの値をとることができ，小数点以上の各桁には上位桁に向かって r^0，r^1，…という重みが付き，小数点以下の桁には下位桁に向かって r^{-1}，r^{-2}…という重みが付きます．

(重み) $r^{n-1}\cdots r^1 r^0\ \ r^{-1}r^{-2}\cdots r^{-k}$

$\downarrow\quad\downarrow\downarrow\ \downarrow\downarrow\quad\downarrow$

$(a_{n-1}\cdots a_1 a_0 . a_{-1}a_{-2}\cdots a_{-k})_r$

$= a_{n-1}\cdot r_{n-1}+\cdots+a_1\cdot r_1+a_0\cdot r_0+ a_{-1}\cdot r_{-1}+a_{-2}\cdot r_{-2}+\cdots+a_{-k}\cdot r_{-k}$

例えば，10進数の $(42.8)_{10}$ は，10^1 の重みが4，10^0 の重みが2，10^{-1} の重みが8となります．

$(42.8)_{10}=4\cdot10^1+2\cdot10^0+8\cdot10^{-1}$

この考え方を使って，ｒ進数から10進数への変換を行うことができます．

$(1011.01)_2=1\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0+0\cdot2^{-1}+1\cdot2^{-2}=(11.25)_{10}$

$(27.36)_8=2\cdot8^1+7\cdot8^0+3\cdot8^{-1}+6\cdot8^{-2}=(23.46875)_{10}$

$(3E9.C)_{16}=3\cdot16^2+14\cdot16^1+9\cdot16^0+12\cdot16^{-1}=(1001.75)_{10}$

表1　ハードウェアに関する出題傾向

出題内容	88		89		90	
	春	秋	春	秋	春	秋
情報の表現形式	⑥⑧	④	①⑩	①	①	②③
論理・シフト演算		④		②	①⑱	⑨
CPU	⑦	③			②	
記憶装置	⑦		③			
補助記憶装置の容量・処理能力	②③	②⑦	②④	③	⑨	⑥
入出力装置	①②	①				
データ通信	⑩			⑩	⑩	⑩

<数字は問題番号を意味する>

■10進→ｒ進変換

10進数からｒ進数への変換は，整数部分と小数部分に分けて行います．

●整数部分（小数点以上）

10進数の整数部分をｒで割り，商と余りを求め，その商をｒで割り，商と余りを求める，という操作を商が0になるまで繰り返します．ここで，余りを逆に並べればｒ進数が求まります．

図1　10進→ｒ進変換（整数部）

```
        余り                 余り
2 ) 42 … 0          16 ) 42 … 10(=A)
2 ) 21 … 1          16 )  2 … 2
2 ) 10 … 0                0
2 )  5 … 1
2 )  2 … 0
2 )  1 … 1
     0

(42)10=(101010)2    (42)10=(2A)16
```

●小数部分（小数点以下）

10進数の小数をｒ進数に変換するには，小数にｒを掛けた結果を整数部(小数点以上）と小数部に分け，さらに，その小数部にｒを掛けて整数部と小数部を求める，という操作を小数部分が0になるまで繰り返します．このとき，掛け算によって得られた整数部分を並べればｒ進数が求まります．

図2　10進→ｒ進変換（小数部）

```
              整数部分
0.4375× 2 = 0 .875
0.875 × 2 = 1 .75
0.75  × 2 = 1 .5
0.5   × 2 = 1 .0

(0.4375)10=(0.0111)2
              整数部分
0.4375×16= 7 .0

(0.4375)10=(0.7)16
```

■2進↔2^m進変換

2進数を2^m進数に，あるいはその逆の2^m進数を2進数に変換するのは簡単です．

例えば，$(11110.01)_2$を8進数に変換してみましょう．$8=2^3$ですから，与えられた2進数を3ビット単位で8進数1桁に置き換えればよいわけです．

$$(11110.01)_2=(011\ 110.\ 010)_2$$
$$=(\ 3\ \ 6\ .\ 2\ )_8$$

一方，$(C5.9)_{16}$を2進数に変換するには，$16=2^4$ですから，16進数の1桁を4ビットの2進数で置き換えます．

$$(\ C\ \ 5\ .\ \ 9\ )_{16}$$
$$=(\ 1100\ 0101.\ 1001\ )_2$$

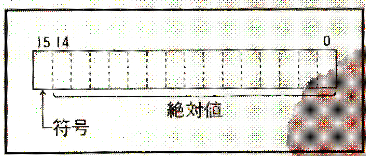
負数の表現

■符号付き絶対値

負数を表現するために，符号を特別なビットを用いて表す方法があります．

例えば，16ビットの領域を，符号を表現する最上位の1ビットと絶対値を表現する15ビットに分けて用います．

図3　符号付き絶対値表示



具体的には次のとおりです．

$(8100)_{10}=(0001\ 1111\ 1010\ 0100)_2$

$(-8100)_{10}=(1001\ 1111\ 1010\ 0100)_2$

■補数

●補数は，数値の中に符号情報を含めて表します．

r進数k桁で表現できる正の数の範囲は0～r^k-1ですが，負の数$-N\ (N>0)$を補数によって，

$$-N\rightarrow N\text{の}r\text{の補数表示}=r^k-N$$

と表せば，k桁で$-r^{k-1}$～$r^{k-1}-1$の範囲の数を扱えるようになります．

例えば，10進数3桁(r=10, k=3)で，−89(=−N)を補数表示してみましょう．

$$-89\rightarrow 10^3-89=911$$

●2の補数は負数を2進数kビットで表現する方法です．定義に従えば，

$$-N\rightarrow N\text{の2の補数}=2^k-N$$

ですが，補数の他の求め方として，

①Nをkビットの2進数で表現する
②各ビットを反転する
③1を加える

という操作を用いる方法があります．

$(-10)_{10}$の8ビット2の補数表示は，

$(10)_{10}=(00001010)_2$ …2進表現
$\rightarrow(11110101)_2$ …ビット反転
$\rightarrow(11110110)_2$ …1を加算

というように求めることができます．

また，この操作の途中でビット反転によって得られた$(11110101)_2$が$(10)_{10}$の1の補数表示になります．

一般に，2の補数を用いたときkビットで表現できる整数値の範囲は，

$$-2^{k-1}\ \sim\ 2^{k-1}-1$$

で，詳しい対応関係は次のとおりです．

<kビット表現>	<数値>
00…00	0
00…01	1
⋮	⋮
01…11	$2^{k-1}-1$
10…00	-2^{k-1}
10…01	$-2^{k-1}+1$
⋮	⋮
11…11	−1

ここで，kビット表現の最上位ビットに注目すると，

最上位ビット＝0　→　正の数
最上位ビット＝1　→　負の数

という関係が成り立つので，結果として最上位ビットが符号ビットと見なせることがわかります．

EXAMPLE 1

(1)負の数の表現に8ビットの2の補数表示を用いるとき，次の2数の組のうち大小関係が正しい組合せを示せ．

ア．$(01110110)_2<(10011000)_2$
イ．$(10000000)_2<(11111111)_2$
ウ．$(-20)_{10}<(11010010)_2$
エ．$(45)_{10}<(00110101)_2$

(2)次の計算結果を10進数で示せ．

ア．$(47)_8+(19)_{16}$
イ．$(2D)_{16}-(31)_8$
ウ．$(C5)_{16}-(305)_8$

解　説

(1)ア．$(01110110)_2=(118)_{10}$
　　$(10011000)_2=(-104)_{10}$
イ．$(10000000)_2=(-128)_{10}$
　　$(11111111)_2=(-1)_{10}$
ウ．$(11010010)_2=(-46)_{10}$
エ．$(00110101)2=(53)_{10}$

したがって，正しいのはイとエです．

(2)ア $=(39)_{10}+(25)_{10}=(64)_{10}$
イ $=(45)_{10}-(25)_{10}=(20)_{10}$
ウ $=(197)_{10}-(197)_{10}=(0)_{10}$

EXERCISE 1

次の数を{ }内に与えられた基数による表現に変換せよ.

ア. $(12.75)_{10}$ {2} イ. $(11.011)_2$ {10}
ウ. $(D.8)_{16}$ {10} エ. $(491)_{10}$ {8}
オ. $(-89)_{10}$ {8ビット2の補数}

EXERCISE 2

次の文章の[ ] 内に正しい数値を入れよ.

(1)負数の表現に2の補数表示を用いると, -100~100の範囲の10進数を表すには[ア]ビット必要である. この形式で$(88)_{10}$は[イ]と, $(-21)_{10}$は[ウ]となる. 一方, $(A8)_{16}$と$(7E)_{16}$は, 10進数でそれぞれ[エ]と[オ]である.

(2)正の小数p (0<p<1) を小数点以下4桁の2進数$(0.b_{-1}b_{-2}b_{-3}b_{-4})_2$で表すことにする. このとき, 0より大きくて最小の数は$(0.[\ ア\ ])_2$, すなわち$(0.[\ イ\ ])_{10}$である. また, 1より小さくて最大の数は$(0.[\ ウ\ ])_2$, すなわち$(0.[\ エ\ ])_{10}$である. また, この形式で, $(0.4375)_{10}$と$(0.24)_8$と$(0.D)_{16}$はそれぞれ, $(0.[\ オ\ ])_2$と$(0.[\ カ\ ])_2$と$(0.[\ キ\ ])_2$となる.

(3)10進数-128を3桁の10の補数で表わすには, [ア]から128を引けばよいので[イ]となる. また, 3桁の9の補数で表わすには, [ウ]から128を引けばよいので[エ]となる.

解 答

● EXERCISE 1

ア. $(1100.11)_2$　イ. $(3.375)_{10}$
ウ. $(13.5)_{10}$　エ. $(753)_8$
オ. $(10100111)_2$

● EXERCISE 2

(1)ア.8　イ.01011000　ウ.11101011
エ.-88　オ.126
(2)ア.0001　イ.0625　ウ.1111
エ.9375　オ.0111
カ.0101　キ.1101
(3)ア.1000　イ.872　ウ.999
エ.871

浮動小数点表示

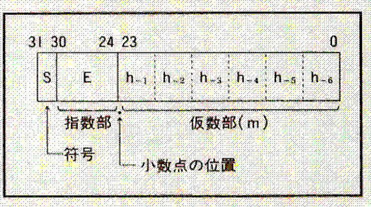
整数を2進数で表現するには, 前項で述べた符号付き絶対値形式, 2の補数形式などによる固定小数点表示方法を用います. 一方, 小数fは,

$$f=(-1)^s \cdot m \cdot 2^e \text{ または } (-1)^s \cdot m \cdot 16^e$$

という浮動小数点形式で表現します. ここで, sは符号, mは仮数(mantissa), eは指数(exponent)と呼ばれます.

いま, s:1ビット, e:7ビット, m:24ビットの32ビット浮動小数点形式において, 仮数部を$m=(0.h_{-1}h_{-2}\cdots h_{-6})$という16進数で表し, 指数部Eにエクセス$2^6$形式(E = e+64)を用いると, 図4のようになります.

図 4　32ビット浮動小数点表示



EXAMPLE 2

$(12.375)_{10}$と$(-0.01953125)_{10}$をこの32ビット浮動小数点表示に変換せよ.

解 説

$(12.375)_{10}$
$=(C.6)_{16}$　…16進変換
$=(0.C6)_{16}\cdot 16^1$…指数のくくり出し

という操作によって, s=0, E=1+64=65, m=$(0.C6)_{16}$となるので,

$$(12.375)_{10} = (41C60000)_{16}$$

が求まります. 同様に,

$$(-0.01953125)_{10}$$
$$=-(0.05)_{16}=-(0.5)_{16}\cdot 16^{-1}$$

と変形できるので, s=1, E=-1+64=63, m=$(0.5)_{16}$となります.

$$(-0.01953125)_{10} = (BF500000)_{16}$$

ここで正規化について説明しましょう. 例えば, 指数のくくり出しは,

$$(0.03)_{16}=(0.03)_{16}\cdot 16^0=(0.3)_{16}\cdot 16^{-1}$$
$$=(0.003)_{16}\cdot 16^1=\cdots$$

というように, 一意に定まりません. そこで, $m=(0.h_{-1}\cdots h_{-6})$の小数第1位$(h_{-1})$が0にならないようなかたち(先の例では, $(0.3)_{16}\cdot 16^{-1}$に相当します)を正規表現として考えることとします.

EXAMPLE 3

この32ビット浮動小数点表示で表現できる数値の範囲を求めよ.

解 説

仮数部は小数点以下の24ビットに対応しますから, 0および2^{-24}~$(1-2^{-24})$の範囲の数を表現できます. 指数部は7ビットですから, -2^{7-1}~$2^{7-1}-1$の範囲の数を表現できます. したがって,

負で絶対値最大: $-(1-2^{-24})\cdot 16^{63}$
→ $(FFFFFFFF)_{16}$
負で0に最も近い: $-2^{-24}\cdot 16^{-64}$
→ $(80000001)_{16}$
正で0に最も近い: $2^{-24}\cdot 16^{-64}$
→ $(00000001)_{16}$
正で絶対値最大: $(1-2^{-24})\cdot 16^{63}$

→ $(7FFFFFFF)_{16}$

となります．

EXERCISE 3

(1)次の10進数を上述の32ビット浮動小数点形式に変換せよ．

ア.35.0　イ.0.03125　ウ.-4.5

(2)この浮動小数点形式で表現された次の数を10進数に変換せよ．

ア.$(41C00000)_{16}$

イ.$(BF400000)_{16}$

EXERCISE 4

次の記述の［ ］内に正しい数値を入れよ．

$\sqrt{2}$ =1.41421356…=$(1.6A09E65\cdots)_{16}$という無限小数は，この浮動小数点表示で［ ア ］となる．同様に，$\sqrt{2}/4$と$4\sqrt{2}$の浮動小数点表示は，それぞれ［ イ ］，［ ウ ］である．

解　答

● EXERCISE 3

(1)ア.$(42230000)_{16}$

イ.$(3F800000)_{16}$

ウ.$(C1480000)_{16}$

(2)ア.12.0　イ.-0.015625

● EXERCISE 4

ア.$(4116A09E)_{16}$

イ.$(405A8279)_{16}$

ウ.$(415A8279)_{16}$

論理演算とシフト演算

論理演算

論理演算はビット単位で演算を行うもので，否定(NOT,$\overline{\ \ }$)，論理和(OR,＋)，論理積(AND,・)が基本となります．さらにこれらを組み合わせて，排他的論理和(XOR)，否定論理和(NOR)，否定論理積(NAND)などが作られます．

EXAMPLE 4

A=$(D5)_{16}$，B=$(F0)_{16}$とするとき，この2数に対して以下の論理演算を行え．

ア.OR　イ.AND　ウ.EOR

エ.NAND

解　説

A＝$(1101\ 0101)_2$

B＝$(1111\ 0000)_2$

ア：$(1111\ 0101)_2$…上位4ビットをオン

イ：$(1101\ 0000)_2$…上位4ビット取出し

ウ：$(0010\ 0101)_2$

エ：$(0010\ 1111)_2$…イの否定

EXERCISE

次の［ ］に正しい数値を入れよ．

8ビットデータの最下位ビットのオン(1)，オフ(0)は，$([\ ア\ ])_2$との論理積をとり，その結果が0でなければオン，0ならばオフと判定できる．また，他のビットを変化させることなく，最上位ビットのみを1にするためには，$([\ イ\ ])_2$との論理和をとる．その逆に，最上位ビットだけを0にするには，$([\ ウ\ ])_2$との論理積をとればよい．

解　答

ア．00000001　イ．10000000

ウ．01111111

シフト演算

左への1ビットシフトは2倍に，右への1ビットシフトは1/2に相当するので，kビットの左シフトは2^k倍に，kビットの右シフトは$1/2^k$に対応します．

最後に，その他の分野に関する出題傾向の要点を述べておきましょう．

中央処理装置

CPUの学習はハードウェアだけでなく，アセンブラ，OSなどの理解にも深く関わる基本的事項です．特に，ハードウェア分野の出題傾向からみると，

- アキュムレータ，インデックスレジスタ，フラグレジスタ，スタックポインタなどの機能・役割
- 命令の実行機構，間接アドレス指定
- 割り込みの種類と要因

などの点を整理する必要があります．

主記憶装置

容量，速度などの単位に注意し，次のような点を整理しておきましょう．

- 半導体記憶装置の特徴，記憶容量，アクセス時間，用途
- 記憶階層化の必要性，方法

表2　論理演算

A	B	NOT $\overline{A}$	OR A+B	AND A・B	XOR A⊕B	NOR $\overline{A+B}$	NAND $\overline{A\cdot B}$
0	0	1	0	0	0	1	1
0	1	—	1	0	1	0	1
1	0	0	1	0	1	0	1
1	1	—	1	1	0	0	0

目から鱗の≪試験直前アドバイス≫

春期の情報処理技術者試験も目前です．そこで，勉強への取り組み方，過去の出題傾向についてもう一度チェックしてみましょう．

意識して関連づけた学習を

皆さんもよくご存知のとおり，2種試験は，午前にハードウェア・ソフトウェアの基礎知識，関連知識が出題され，午後にプログラムの作成能力を問う流れ図，言語に関する問題が出題されます．出題分野はこのように分類できますが，これらの分野は各々独立しているわけではなく，互いに関連しています．したがって，分野ごとにバラバラに割り切って勉強するのではなく，他の分野との関係を常に意識しながら学習することが大切です．例えば，ハードウェアに関する情報表現，論理演算，CPUなどの分野は，CASLに深く関わってきますし，ソフトウェアに関するアルゴリズム，データ構造，ファイル編成などの分野は，流れ図，言語の学習にも通じるところがあります．さらに，日頃から各々の分野で使われている英語の技術用語を，“音”としてとらえるのではなく，省略形ではない“ことば”として理解することに努めれば，英語の学習と一石二鳥になることは言うまでもありません．

過去の出題傾向を押さえておこう

午前の問題

●ハードウェアの基礎知識

ハードウェアはこの後で述べるソフトウェアと比べて，毎年ほぼ似かよった内容の問題が出題されるので，ポイントをしぼった学習ができます．

主な出題内容は次のとおりです．

①情報の表現，論理・シフト演算
②CPU，主記憶装置，入出力装置
③補助記憶装置の容量・処理能力
④データ通信

このうち，①，③は計算問題で，ほぼ毎年出題されています．計算問題というと敬遠されがちですが，解き方・公式を理解してしまえば，他の“暗記もの”と比べて，かえって点を稼ぎやすくなりますから，しっかりと勉強しておきましょう．

●ソフトウェアの基礎知識

最近の出題内容は次のとおりです．

①データ構造とアルゴリズム
②ソフトウェアの開発
③OS
④ファイル
⑤情報処理システムの構成・特徴

①は整列，探索などのアルゴリズムを単独で扱うのではなく，配列，リスト，木などのデータ構造と関連づけて学習することが重要です．また，実際に流れ図，プログラムを作成すれば，午後の問題への対策にもなります．

②はソフトウェアの開発手法，開発環境に関するもので，最近よく出題される傾向にあります．⑤は，ほぼ毎回出題される分野で，データ通信，信頼性の計算などと関連させた内容となることが多いようです．

●関連知識

数学，工業，商業，英語という分野から，各々2，3～4，3～2，2問という割合で合計10問出題され，その中から5問を選択解答します．数学は高校の教科書レベル，工業はOR，在庫管理，工程管理，電気・電子工学，物理などの分野，商業はほぼ簿記，会計からの出題と考えてよいでしょう．英語では，情報処理分野で話題になっている内容の出題が多いようです．

学習の対策として，4つの出題分野のうち少なくとも1つを得意分野としておくことが大切です．

午後の問題

●流れ図

基本的なアルゴリズム，事務処理，技術計算という3問の中から1問を選択します．この分野は他との関わりも深く，大変重要です．学習にあたっては，流れ図を目で追ううわべだけの勉強で終わるのではなく，実際に手を使って具体的にトレースすることが大切です．選択解答ですから，少なくとも1つの得意分野を作るように心がけましょう．

●言語

CASL，COBOL，FORTRAN，PL/Iの4問中1問を選択解答します．プログラミングの学習は，まず手を動かしてプログラムを作ることです．そして，人の作った様々なプログラムをトレースすることも大切です．“習うより慣れろ”の精神で取り組みたいものです．1問選択だからといって，1つの言語だけにしぼらずに，今後のステップアップを考えて，1種で必須のCASLと他の1言語のマスターを心がけましょう．

合格のためには計画的な学習が大切です．とはいえ，春期の試験は目前です．「今からでは間にあいそうもないので，ここはイッパツ，ヤマを懸けよう」と考えている人もいるのではないかと思いますが，そのような人には，“自分の弱点を知る”ことをお奨めします．過去の問題をやってみて，“自分が分からない分野は何か？”，“どういう形式の問題が苦手なのか？”という点をしっかり把握して，そこを重点的に勉強するとよいでしょう．

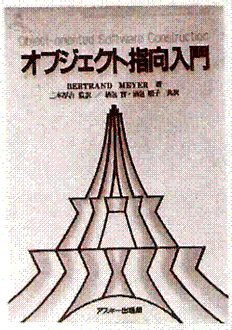
Book Review

オブジェクト指向の本質に迫る議論がみっちり展開されている

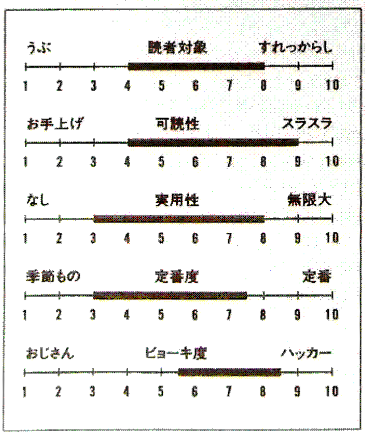
■中村正三郎

■オブジェクト指向入門

Bertrand Meyer著
二木厚吉　監訳，酒包寛・酒包順子　共訳
アスキー出版局　A5判　640ページ　4,980円



■評価チャート



本書の性格は，監訳者まえがきにおいて明確に述べてある．

「オブジェクト指向という言葉は使う人によって」異なり，「本書におけるオブジェクト指向もメイヤーさんの『オブジェクト指向』である」．しかし，「本書には，ソフトウェア開発における最も系統的かつ効果的な方法の原理と，Eiffel言語を用いたそれの具体例に良質の解説が，完結した形で含まれている」．

メイヤー氏はオブジェクト指向による設計，開発，保守がいかに優れたものであるか，また優れたものとするには，設計開発者は，どのような態度でソフトウェア開発に立ち向かうべきか，またそれを支援する言語，環境はどうあらねばならないかを，本書を通じて力説する．原書の書名はそのものずばり“Object-oriented Software Construction”である．この意味において，本書の書名「オブジェクト指向入門」は，いささか正しくない．入門という言葉から連想される俗な解説ではなく，オブジェクト指向の本質に迫る議論が広い範囲にわたって，みっちり展開されているからである．

本書は大きく4つの部分に分れる．

Part 1「問題と原則」では，高品質のソフトウェアに求められる要因を分析し，ソフトウェア開発における問題点と，それを解決するためにソフトウェア工学，各種プログラミング言語では，どういうアプローチがなされてきたかを明らかにする．この考察によって，オブジェクト指向アプローチが広く深く有効であることが示される．

Part 2「オブジェクト指向による設計とプログラミングの技法」では，Part 1で述べた議論を，自ら設計開発したEiffel言語を用いてより具体的に展開する．Part 1が抽象論ならここは具体論である．Eiffel(英語で読めばアイフル）は，彼が考えるオブジェクト指向によるソフトウェアの設計開発を，支援するための言語である．オブジェクト指向の概念説明とともにEiffelを紹介している．

Eiffelは表明（assertion）を前面に押し出している点で特徴的だ．形式的仕様記述の重要性は，かねてから指摘されているが，プログラミングの現場ではいつも無視されてきた分野なのだ．メイヤー氏には，プログラミングの理論（代数的意味論，公理的意味論や表示的意味論の世界）の著作もあり，これがこの言語の設計の背景にある．

Part 3「そのほかの環境におけるオブジェクト指向法の応用」は，Pascal, Fortran, Cなど古典的言語，カプセル化を提供するAda，各種オブジェクト指向言語Simula, Smalltalk, C++, Lispなどにおけるオブジェクト指向のあり方を示している．これはまた，Eiffelがなぜ現在の言語仕様になったかの説明にもなっている．

Part 4「付録」は，Eiffelの言語仕様の記述である．特に付録Aの「Eiffel

ワンラインNEWS
ロータスは富士通のノート型パソコン「FM R-NB1」「FM R-CARD」に対応したICカード版の「Loutus 1-2-3R2.2J」を発売．「Loutus 4 word」も同梱されている．

ライブラリからの抜粋」は，この言語でのコーディングスタイルがどんな姿になるのかを，よく映している．

本書はプログラミングの初学者にはつらいだろう．目配りはしてあるが，やはりソフトウェアの設計開発に対する基礎知識や経験は，必要だと思う．だが，大部の書ではあるが主張は明確である．オブジェクト指向設計を試みている人にも，概念の整理に役立つ．

最後に苦言をひとつ．参考文献に挙げられているものの中には，邦訳が出ているものがある．後学の便宜のために，邦訳があるものはそれを示してほしかった．

ますます好調/91年はオブジェクト指向の年に

古田島義和（神田・書泉グランデ）

オブジェクト指向の世界がにわかに注目を集めはじめてきました．91年に入ってから，「オブジェクト指向のプログラミング」（トッパン），「オブジェクト指向・システム分析」（啓学出版），「オブジェクト指向入門」（オーム社），「C++オブジェクト指向プログラミング」（トッパン），「プログラミング言語C++」（トッパン）などを中心に動きはじめています．しかし，上記の作品のほとんどが訳本という現状を無視できません．まだまだこの世界は，米国の洋書に頼るしかないというのが現状です．

91年度オブジェクト指向関連書籍のベストセラーと新刊を紹介すると，まず「Object-Oriented Analysis」（Prentice Hall／以下PH，Coad著）があげられます．この本は2/Eが出版され，初版と同様好調な売れゆきです．今回「The C Users Jounal」（91/Feb）に書評として載った，「Object-Oriented Design with Application」（Addison Wesley／以下AW，Booch著）も売れています．

オブジェクト指向の中心的な存在に，Meyer氏を忘れてはなりません．代表作に「Object-Oriented Software Construction」があり，これをアスキーが「オブジェクト指向入門」として出版し，人気を集めています．この他に，「Annotated C++ Reference Manual」（AW，Stroastrup著），「C++ Primer」（AW，Lippman著），「Inside Smalltalk Vol.1&2」（PH，LaLonde著），「Teach Yourself... C++」（MIS），「Turbo C++ Disk Tutor」（McGraw Hill），「Object - Oriented Programming in Turbo Pascal 5.5」（AW）などが好調です．

新刊では，「An Introduction to Object - Oriented Programming」（AW，Budd著），「Object-Lifecycles: Modeling the World in States」（PH，Mellor著），「Research Foundations in Semantic and Object - Oriented Databases」（PH，Cardenas著），「Object-Oriented Databases with Applicaitons to CASE, Networks, and VSLI CAD」（PH，Gupta著），「Object - Oriented Databases」Hughes著），「Object - Oriented System Analysis and Specification:A Model Driver Approach」（PH，Kurtz著），「Object-Oriented Modeling and Design」（PH，Runbaugh著），「Designing Object-Oriented Software」（PH，Wirfs-Brock著），「C++:A Guide for Programmers」（PH，Hekmatpour著），「A First Course in Smalltalk-80」（PH，Hopkins著）と話題性のあるものを取り上げてみましたが，まさに，オブジェクト指向の年のように感じます．既に米国では，オブジェクト指向専門雑誌「Journal of Object-Oriented Programming」と「C++ Journal」の2誌があります．

洋書のベスト10を見ると，訳本が出版されている（「コンパイラ I，II巻」（サイエンス社））にもかかわらず，「Compilers」（AW，Aho著）が堂々第1位になっています．定価4千円弱というのが人気の秘密？なのでしょうか．この著者とUllman氏のコンビは，これまでアルゴリズムとデータ構造などか数々の名著を執筆しましたが，両氏の「Foundations of Computer Science」（Computer Science Press）が近日入荷予定です．ベスト10をネットワークとGUIが占める中，4位の「Undocumented DOS」（AW）が注目を集めています．

■和書&洋書ベスト10

（書泉グランデ調べ：1991年1月4日～1月31日）

1	rootから／へのメッセージ	アスキー	Compilers (paper)	Addison Wesley
2	コンサルタントの秘密	共立出版	Simple Book:An Introduction to Management of TCP/IP	Prentice Hall
3	VZ倶楽部	ビレッジセンター	Programming Windows 2/E	Microsoft Press
4	MS-Windowsとは何か	講談社	Undocumented DOS	Addison Wesley
5	たのしいUNIX	アスキー	Mac Scheme:Manual & Software	M.I.T Press
6	C言語を256倍使うための本	アスキー	Inside Smalltalk Vol.2	Prentice Hall
7	VZを256倍使うための本	アスキー	System Performance Turning	O'Reilly
8	入門C言語　改訂新版	アスキー	OSF/Motif:Programmer's Guide Releases 1.1	Prentice Hall
9	X68000マシン語プログラミング　入門編	ソフトバンク	UNIX Network Programming	Prentice Hall
10	UNIXユーティリティライブラリ　make	啓学出版	Computer Networks 2/E	Prentice Hall

佐々木哲のニューヨーク N.Y. システム通信

速いモデム！

いやー，高速モデムが安くなりましたね．9600bps全2重のV.32モデムは数年前までは数千ドルしていたのが，今では500ドル以下．VGAボードでおなじみのATI[*1]や，今ではHayesの一部門であるPractical Peripherals[*2]がV.42bis付きのV.32モデムを定価699ドルで販売を始めました．実売価格は400ドル台でしょう．一気に9600bpsユーザーの増加が見込まれます．どちらもV.42bis付きです．

USRobotics独特のHSTモデムを使ってBBSにアクセスするだけの時代もありましたが，もうそんなこともありません．Compuserveも正式にV.32による9600bpsのサービスを開始しました．その一方，Hayesをはじめ代表的なモデムメーカーは14400bps全2重のV.32bisに移りつつあります．え？　ISDN？？　あまりUSでは聞かないです．日本のように個人で設置する人は聞いたことがない．

＊1:ATI Technologies Inc., 3761 Victoria Park Ave., Scarborough, Ontario M1W 3S2, phone (416) 756-0718, fax (416) 756-0720
＊2:Practical Periphrals, Inc.,31245 La Baya Drive, Westlake Village, CA 91362, phone (818) 706-0333, fax (818) 706-2474

より高速に！NS16550AとHayesのESP

高速になるにつれ，PC本体側の負荷が高くなってきます．CPU性能の低い機種，ましてやDesqViewやMS-Windows上でバックグラウンド処理中にダウンロードさせたりすると，データの取りこぼしは必至！そもそも，PC互換機のIntel 8250UARTに頼ったシリアルインターフェースは安価ではありますがもう性能的に限界です．いくつか解決策がでてきました．

まず，このシリアルチップ自体の置き換えです．National Semiconductorの16550Aという8250上位互換，ピン互換のUARTがあります．高速な非同期通信をするために，いわばAT互換機のために開発されたチップです．IBM PS/2でも正式に採用されています．これは内部の16バイトほどのFIFOバッファを持っています．これをサポートした通信ソフトを使えば，マルチタスクのきびしい環境下でも安定して高速でデータの送受信ができます．多くの通信ソフト，たとえば広く使われているシェアウェアのTelixもこれをサポートしています．オリジナルのクロック4.77MHz8088CPUのIBM PCでも大丈夫．これから機械を購入するときは，これが使われているかちょっと見てからにしましょう．あるいは最悪8250がソケットに挿さっていれば，あとでNS16550Aを購入して差し替えることも簡単です．チップは20ドル以下で入手可能．

ひとつ困りものがいます．MS-Windows 3です．この威力を一番必要としているはずのものがサポートしていないのです．Windows下で通信ソフトを動かすと，Windowsが I/Oも面倒を見ます．今のところ，NS16550Aの付加機能がサポートされていないので，上のTelixであろうとWindows専用通信ソフトであろうとただの8250を使っているのと同じなのです．これはMicrosoftのサポートを待つしかありません．

とはいえ，現在の1バイトのデータを取り込むために，数百，数千バイトの割り込みルーチンが駆動されるやり方ではCPUの負荷が増えるばかりです．PCの通信スタンダードをリードしてきたHayes[*3]は一歩進めました．Enhanced Serial Interface，略してESIという規格です．ESPという2つのNS16550A UARTとCPUが載ったボードを販売しています．これは単なるNS16550A付きシリアルポートとしても使えますが，専用のESPドライバを通して使えば，いままでとは比較にならない高速でバッファ付き，DMAサポートあり，ブロック転送可の高速インターフェイスとして使用できます．インテリジェントなI/Oだから，どんなに高速な通信をしてもCPUにはそれほど負荷はかかりません．今のところDOS，Windows，OS/2でドライバがサポートされています．誰でもこのインターフェースを使って開発できます．このボードは値段が300ドル程度です．

Vシリーズモデムで紹介された，非同期モデムでホストとの同期通信を可能にするAutosyncをはじめ，的確に標準をリードするHayesの姿勢はたいしたものです．

＊3:Hayes Microcomputer Products, Inc., P.O. Box 105203, Atlanta, GA 30348,　(800) 635-1225

FAXボード，WindowsのFAXドライバも

通信とPCというのはとても相性のいいものです．その中でもPC用のFAX関連製品はとどまるところを知らずに広まっています．PCからFAXを送るにはFAXモデムが必要です．V.27terあるいはV.29のCCITT規格にあったモデムに加えて，それをコントロールするソフトウェアが必要です．すでに市場には，100種類以上の実に多様な製品が出ています．

ごく安いものは，通信速度も4800bpsまでしかサポートせず，送信専用と割り切っています．またダムなモデムのため，専用のソフトを起動している間しか送受信できなかったり，メモリに常駐させ他のソフト使用中に送信するにしても，CPUを占領してしまうもの．メモリ常駐サイズも200KBを越えるもあり大きめです．

高級なものではボード上に専用のCPU，メモリを持ち，本体のCPUに負荷をかけずに完全にバックグラウンドで9600bpsの送受信ができます．また，比較的メモリ常駐量も小さめです．

使い方もいろいろ．一般的にはファイルを指定して送信させます．形式はテキストファイル，代表的なワードプロセッサのファイル，またPCX，TIFFなどのグラフィックファイルです．受信したFAXは，普通そのFAXボード固有の形式でグラフィックファイルとしてディスク上にセーブされます．また，望ましい機能としては，プリンタのエミュレーションがあります．ふだん使っているアプリケーションから印字したものが，そのままFAXとして送られる．より複雑なためか，メモリ常駐サイズの増加を招きます．さらに印字したものを，そのまま前もってチェックせずに送ってしまうのも心配です．かといって機能を増やせば，さらにサイズが大きくなり使いものにならなくなります．

これに対するすばらしい解決策があります．MS-Windows用のFAXドライバです．プリンタドライバとしてWindows下で動作し，プリンタに印字する代わりにFAXボードに送るのです．MS-Windows下ではプリンタのインターフェースが一元化されていますから，1種類のドライバで，すべてのWindowsアプリケーションをサポートできます．Ami Professional，PageMaker，Excelなどなど，印字するだけですぐにFAXとして送れます．ドライバ自身Windowsアプリケーションですから DOSのようなメモリ常駐量の問題，他のソフトとの相性の問題もほとんどありません．

現在これを実現した製品が2つあります．Alien ComputingのFaxit!とDelrina TechnologiesのWinfax[*4]です．前者はIntel PCEOとタイアップして低価格で販売され一躍有名になりました．後者は安価なSierraのSendfaxチップを使ったFAXボードをサポートします．

究極のFAXボードというと，Intelの新製品SatisFAXtion[*5]です．Connection CoProccessorの後継で，PC/ATバス用は実売価格400ドル以下です．マイクロチャネル用のものもあります．ボード上には9600bpsの送受信FAXに加え2400bpsのMNPモデム付き．80188 CPUに128KBのメモリが載っていて，完全にバックグラウンドで動作します．そのためパソコン側のCPUにはほとんど負荷がかかりません．ドライバのメモリ常駐サイズはたったの13KB．どんなに重いソフトを動かしていても気づかないほど，バックグラウンドで送受信します．購入すると上記のFAXit!が19ドルで買えるクーポン券が付いてきます．MS-WindowsでATMやFaceLiftなどのフォントユーティリティとともに使うと，PostScriptプリンタでの印字かと思うほど美しいFAXを送ることができます．

＊4:Winfax $69.00，Delrina Technology，phone (416) 441-3676
＊5:SatisFAXtion $499.00/$549.00 (MCA)，Intel Personal Computer Enhancement Operation，5200 NE Elam Young Pkwy.，Hillsboro，OR 97124，phone (503) 629-7354.

今月の何だこりゃ？

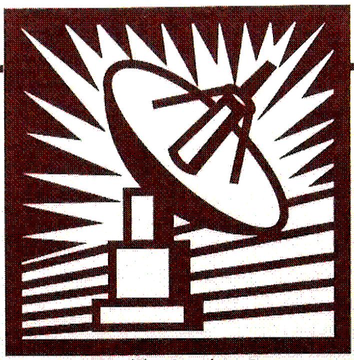
もひとつ，FAX関連！

パソコン用のFAXボードの苦手は受信です．ずっとパソコンの電源をいれておく必要があります．またパソコンで受信したFAXをプリンタで印字しなおすのも面倒な話．いっそ，プリンタをFAX受信機にしてしまおうという製品があります．その中でももっとも安価なものをひとつ．周辺機器の老舗Tall Tree Systems[*6]のFax-O-Maticは，HP LaserJet IIや，互換のページプリンタにパラレルポートで接続する受信専用FAXモデムです．内部に512KBのバッファを持つため，通常の印字処理のじゃまにはなりません．この399ドルの製品で安価な普通紙FAXのできあがり！

＊6:Tall Tree Systems，2585 E. Bayshore Rd.，Palo Alto，CA 94303，phone (415) 493-1

まだまだコンピュータと通信，FAXの使い方はいろいろあります．乞うご期待．

Illustration : NOBUKI FUKUHARA



西川和久の

希望的観測

日本語Windows 3.0, 第1弾はNECから登場!!
はたして日本では米国ほど普及するのか!?

1 待ちに待った日本語Windows 3.0 走行環境・パフォーマンスは?

昨年5月22日に米国で大々的に発表され, かつてない成功を納めたWindows3.0の日本語版がやっと登場した. 大筋の予想通りNEC版からである. 他社からも順次発表・発売されるであろう. 全体的に見ると日本語版として登場するまでに, およそ9か月から12か月かかった計算になる. この期間が早いか遅いかという論議や, 日本語Windows 3.0自体の機能などは, またの機会にお話することにして, 今回は「日本語Windows 3.0は米国ほど普及するか!?」に焦点を絞ってみようと思う.

私は, 日本語Windows 3.0が一般化するために, いろいろ通過しなければならない関門があると思っている(日本語Windows 2.1のころから私が問題点として指摘してきた点も含まれているが……). これから順番にポイントを押さえてみよう.

一番はじめに気になるのは, 走行環境とパフォーマンスであるが, 早くもつまずいてしまう. 必要メモリ容量である. メーカーによって多少異なるが, 平均すると「標準モード3MB以上」「拡張モード4MB以上」と考えてほぼ間違いないであろう(もちろん, これ以上のメモリを搭載しているほうが良いに決まっている). パワーユーザーならいざ知らず, 表計算ソフトやワープロソフトをおもに使っているユーザーで, ここまでメモリを拡張しているとは思えない. 少なくとも2MB以上のメモリ追加になるであろう. また, テキストを主とするアプリケーションとくらべると, スクロールなどの基本描画速度がお世辞にも速いとはいえない. 当然マシンパワーも並以上が要求される.

日本語Windows 3.0の数ある機能と代償に, これだけのコストをかける必要性がどれだけあるのかは疑問である. いろいろ便利になるとはいえ, これまでのDOS環境でさほど不自由していないのだから…….

2 対応アプリケーションの数と値段? 画面サイズは2.1以来の課題

米国での状況を見てみると, 現在およそ300本程度のWindows 3.0アプリケーションがあるといわれているが, こと国内になると桁が1つ少なくなってしまう. また気になる値段も, 5万円未満のものなどほとんどないといってもいい. ビギナーがおいそれと手を出せる範囲ではないのだ. Windowsに付属する標準アプリケーションでもある程度のことはできるが, それだけではなんの意味もない.

これは, Windows市場の熟成を待つしか手がないのだが, 「鶏と卵」にも似た関係で, なかなか難しい問題である.

画面サイズの件は, 日本語Windows 2.1の時代から私が問題点としてあげていることだ. 日本語GUI(グラフィックユーザーインターフェース) を考えた場合, 640×4?0程度では画面が非常に狭いのである. 狭い画

ワンラインNEWS
米国側が強い競争力を誇っていたコンピュータ関連の貿易収支がいよいよ赤字になった. 周辺機器の輸入急増が原因といわれる. 来年はさらに増える見込み.

面にわざわざ窓を開く必要性がどれだけあるのだろうか!? 高解像度になれば良いのだが,国内には「高解像度マシンは高価である」という風習がある.最悪なのは(一部例外はあるが),標準解像度マシンから高解像度にしたければ,マシンやソフトウェアごと全取り替える必要があることだ!! オプションとしてカードなどを追加するだけで対応できるよう,各メーカーサイドの方に強く要求したい.

3 DOSアプリ,デバドラなどとの相性?そしてフォントなどの問題

Windows 3.0は,従来のWindows/386の機能を含んだものになっている.日本語Windows/386では,80386の仮想86モードを使って複数のDOSアプリケーションを起動することができた.そのときに経験したのだが,普段使っていたデバイスドライバ,たとえば,ディスクキャッシュやRAMディスクドライバなどのほとんどが使えないのである.

これではいくらタスクスイッチャーがわりに使えるとはいえ,アプリケーションを実際に使う場合のトータルパフォーマンスが落ちてしまう.また,不確定要素でのハングアップも数多く経験した.こんなことでは安心してタスクスイッチャーがわりには使えない.その点は日本語Windows 3.0でどうなっているのだろう.気になるところである.

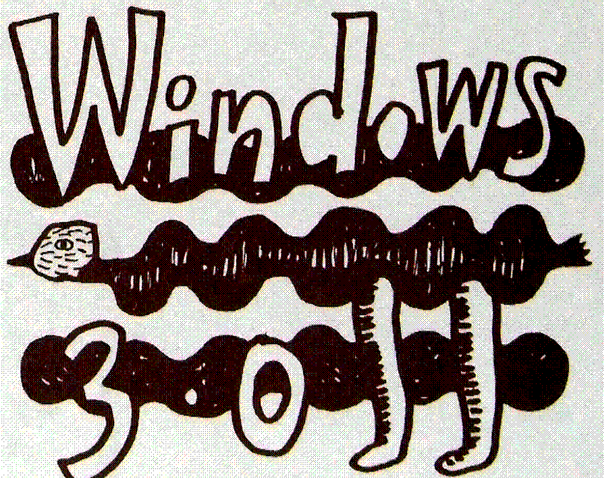
拡張モードでは,テキストベースのDOSアプリケーションを窓上に展開して動かすことができ,表示データのカット・ペーストが可能となっているが,「可能」と「使いものになる」とでは大きなギャップがある.このへんのことが「使いものになる」レベルで仕上がっているのだろうか?

「プラットフォームとして,異機種間で同じアプリケーションを動かす」ことは,Windowsの特徴としてよくいわれるが,国内では現実問題として「××専用」となっているパッケージが多い.アプリケーション側の問題(?)なのか,対象となるメーカーのWindows本体の問題(!)なのか理由はよくわからないが,これでは,Windows自体の存在価値が半減してしまう.マニュアルの記述などいろいろあるのだろうが,なんとか解決していただきたい.

米国版Windows 3.0では,数多くのフォント・文字サイズなどが標準搭載され,Adobe Type Managerなどに代表される,汎用的なスケーラブルフォントも単品で販売されている.当然ながら,日本語Windows 3.0を実用的に使うには豊富なフォント・文字サイズが必要になってくるが,容量的な問題からシステムであらかじめ準備されているものしか使えない.日本語では,データ量や展開速度など技術的問題があることはわかるが,できるだけ早い時期に解決して欲しい.

4 絶望的観測(!?)が希望的観測になるには……

私なりの考えを書いてみたが,結果的に「はたして日本語Windows 3.0は米国ほど普及するか!?」という表題に対して,否定的なものばかりになってしまった.これでは,絶望的観測だ.残念ながら,これが国内における現実である.これらの条件がある程度満たされないと,話題性から興味半分で買った日本語Windows 3.0も,結局のところ使われずに眠ってしまう可能性がある.私の予想を覆し,日本語Windows 3.0が普及してくれれば非常にうれしいのだが…….読者の皆さんはどのように思われますか!?



コンピュータの法律

市販のプログラムのコピーは自由か

◆

熊谷信太郎(弁護士)

今回は，前回に引き続いて，プログラムと著作権に関する具体的な問題を検討したいと思います．

■原則的には著作権を侵す行為

市販のプログラムを購入した後で，自分でバックアップ用のコピーを作成したり，友人に頼まれてコピーしてあげるということは多く見られますが，これは著作権法上どのような問題があるのでしょうか．

著作権法では，プログラムそれ自体を著作物として保護していることは前回までに説明しましたが*1，著作権とは英語でCOPY RIGHTと表示することからもわかるように，もともと，COPYすなわち複製されない権利として構成されてきました．したがって，プログラムを著作権者に無断でコピーすることは，原則的に著作権を侵す行為であって，許されません．

■自分で使用する限度において

しかし，一方でプログラムというものの性質上，摩滅や棄損に備えて，バックアップ用のコピーをとっておく必要性が高く，実際の取り扱いも，そのようにしている人が多いという現実がありました．

それにもかかわらず，バックアップ用のコピーをとる場合，そのつど著作権者の許可を得なければならないとすると，煩雑で面倒です．

そこで，著作権法は，市販のプログラムを購入した人などは「自分で使用する限度において」，プログラムのバックアップ用のコピーをとることができると定めました*2

ただ，これはあくまで「自分で使用する限度」においてコピーに関する制限を解除するというものですから，当然，コピーすることができる範囲は限られてきます．たとえば，友人に分けるためにコピーすることは許されませんし，また，家庭内で使用するためにバックアップ用のコピーをとる場合でも，「自分で使用する限度」においてという制限がありますから，大量にコピーするようなことはできません．

■法人に関する2つの論議

これに関して，会社などの法人が1枚のプログラムを買ってきて，社内で大量にコピーして各セクションで使うという行為は，複製権を侵害する著作権侵害ではないかという点が，最近議論されています．

これについて，著作権侵害ではないとする立場の主張は次のような内容です．すなわち，個人がプログラムを買ってきて自分で使用する限度ではコピーができるとされている．それならば，同様に，法人が買ってきたプログラムを自分で使用する限度でもコピーができるはずである．そして，法人というのは内部的には各セクションで構成されているのだから，法人の各セクションがプログラムを使うのは，まさに，法人が自分で使用する行為にほかならない．したがって，各セクションが使用するためにプログラムをコピーするのは許される，と主張しています．

これに対して，そのような行為は著作権侵害であるとする立場は，プログラムのコピーは本来は著作権者の許可を得なければならないものであって，それを「自分で使用する限度」において制限的に解除しているのであるから，バックアップのためのコピーもあくまで，必要最低限の枚数に限るべきである．そして，個人がバックアップ

Illustration : Nobuki Fukuhara

ワンラインNEWS
米IBM社は自社のプリンタ部門を売却する．同社は昨年キーボード部門などの情報関連機器を分離した．売却してもブランド名は変わらない．

のために大量コピーすることが許されないのであるから，それと同様に，法人もいくら各セクションで使用する必要があるとはいっても，コピーは必要最低限の枚数しかできないというべきであり，各セクションでプログラムが必要ならば，必要とされる枚数のプログラムを購入すべきである，と主張します．

この問題については，まだ法律的な判断が固まっていませんが，現在のところ，「自分で使用する限度」においてのみ，必要最低限のコピーが許されているという著作権法の趣旨からすると，社内で大量にコピーして各セクションで使うというのは，著作権侵害の可能性が高いという判断が一般的になりつつあるといえるでしょう．

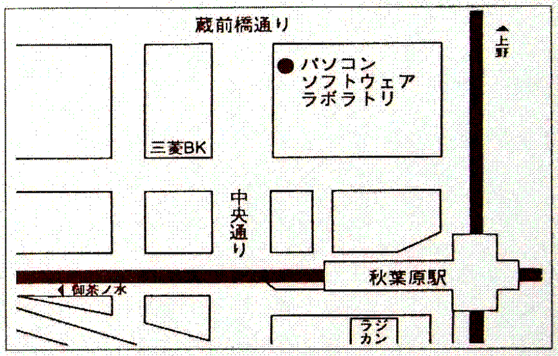
＊1　著作権法第10条
＊2　著作権法第47条の2
プログラムの著作物の複製物の所有者は，自ら当該著作物を電子計算機において利用するために必要と認められる限度において，当該著作物の複製又は翻案をすることができる．

アキバ短信

パソコンソフトウェアラボラトリ

1600本を超えるソフトが無料で使える!?

住所　東京都千代田区外神田4-7-1　NEC秋葉原C&Cセンター
TEL　03-3254-9211（代）
営業時間　10:00〜18:00　　休日　月1回(不定期)



秋葉原というと大型電機店の派手なディスプレイに惑わされがちだが，実はメーカーのショールームも多い．読者の中にはパソコン歴？年，いまさらショールームなんかへ行く用なんぞあるものか！とお思いの向きもあるかと思うが，今回紹介するソフトウェアラボラトリはNECのPC-8800，9800シリーズ用ソフトウェアのショールームなのだ．

ここでは1600本あまりのソフトが，ワープロ，データベース，教育用，OS／言語などのジャンル別に分類，管理されている．使ってみたいソフトがあれば，ラボラトリ内のパソコンで自由に試用できるという仕組みだ．

まず，カウンター横の端末を使って，目的に応じたソフトを検索する．内容が知りたければカタログを見ることもできる．次に，備え付けの用紙に試用したいソフト名，住所，氏名などを書いて申し込む．係の女性がフロッピーとマニュアルを出してくれるので，指定されたブースのパソコンでこれを利用する．

ラボラトリ内に設置されているパソコンはPC-9800シリーズが11台，PC-8800シリーズが2台だが，そのほとんどがプリンタ付．中には大容量のハードディスクを備えた32ビットマシンやA4フルサイズのイメージスキャナ，スピーカー付のものもあって，ワープロや業務用ソフトに限らず，WindowsやOS/2，お絵描きソフトや音楽ソフトも試用できる．希望すればノートパソコンも貸してくれるそうだ．試用できるソフトウェアの一覧は定期的にPC-VANの「情報ファイル」にアップしており，電話での問い合わせにも応じてくれる．

訪れる人は実にさまざま．業務用のソフトを探しに来るビジネスマン，今はやりのWindowsとはどんなものかを見にきた学生さん，通りすがりにフラリと立ち寄る人もあるという．ただし，ソフトの試用を目的とした場所だけにソフトの説明などはなし．ソフトの持ち込み，持ち出し，コピーなどができないよう，ソフト試用中はロッカーに荷物を預けることになっている．

気にいったソフトがあれば，カタログライブラリからカタログをもらって帰るもよし．一歩外へ出ればそこは秋葉原電気街，その足で買いに行くのもまたよし．利用するなら比較的すいている午前中がおすすめだ．

ワンラインNEWS
日本の半導体メーカーがEC委員会と結んでいる輸出最低価格がDRAMだけでなくEPROMにも適用されることになった．将来の輸出障害にならなければいいが．

新連載●プログラマの仕事場 | 1

【PW】 Programmer's Working Space

株式会社ソフトヴィジョン

プログラマのプログラマによるプログラマのための会社

ソフトヴィジョンという会社名を知らない人も，本誌Book ReviewやThe BASIC誌「電脳騒乱節」などでおなじみの中村正三郎氏や，オンラインソフトKI-Shellの作者石野恵一郎氏の名前は聞いたことがあるだろう．中村氏も石野氏もソフトヴィジョンのメンバーである．ソフトウェア業界では，知る人ぞ知る高い技術力を誇る会社なのだ．

誇り高い「プログラマ」集団

社長の酒井氏は，同社を「プログラマが集まってできた会社」と呼ぶ．一般に「プログラマ」という言葉は，システムエンジニアの設計した仕様に基づいてコーディングとデバッグを担当する一段レベルの低い仕事という，ネガティブな意味あいが込められることが多い．会社によってはこれを嫌って，ソフトウェア技術者をすべて「システムエンジニア」という呼び名で統一しているところもあるほどだ．あえて「プログラマ」という呼び名にこだわるのは，品質の高いソフトウェアを作成するには，設計する力とそれを作り上げる力がともに重要であると考えているからだという．

酒井氏自身，かつてデータベースソフト「桐」の開発に携わった経験をもつ．

「パソコンのソフトは，最近では結構しっかりしてきましたが，おもちゃみたいなものが多い．きちんとしたソフトウェア科学に根ざして作られているものが少ない，あるいはそうしようと思っているけれども，いい加減に作られている．「桐」は，『スピードが速い』『いろいろな機能が充実している』という評価を得ましたが，ソフトウェア工学のしっかりした教育を受けた者から見れば，特別なことは一切やっていません．アイデアとして『データベースをスプレッドシートのインターフェースで』という発想はありましたが，当り前のことを当り前にやっているだけです．業務で使う人が本当に楽になるソフトウェアを作りたいと思っています」という．

「プログラマの作った会社ということで，実際にソフトウェアを作るのがわれわれの仕事だという意識が強いですね．アメリカのようにソフトウェアを作って，それがユーザーに評価されて広まっていく，そういう形が理想ですね．本当にプログラムが好きで，プログラムを作る仕事がしたい．『作ってなんぼ』の世界ですね（笑）」．

開発は自社，販売は他社

ソフトヴィジョンでは，どんなソフトウェアを開発しているのだろう．同社が開発したパソコン用のパッケージソフトには，MS-Windows用の日本語入力ユーティリティWinATOK，通信ソフト

ワンラインNEWS
キヤノン販売系のキヤノンソフトウェアは経理の仕事をパソコンで学べる「ハイパーCAI経理入門」を発売した．対応機種はAppleのMacintosh．

MILINKなどがある．これらのソフトウェアは，ソフトヴィジョンのブランド名では販売されていない．その理由について酒井氏は次のように語ってくれた．「まずはソフトウェアの開発がやりたいということです．プログラマばかり集まってしまったので，販売ができるスタッフがいない．販売のできる組織にするつもりも今のところはありません．当面は，提携した会社に販売をお願いして，開発に専念したいですね．実際に，私たちが企画・開発したソフトを提携会社から販売する予定で進んでいるプロジェクトもあります」．

自主開発と受託業務半分ずつが理想的

プログラマ本位，ソフトウェア開発中心というのがソフトヴィジョンの方針だが，実際に仕事をしていくうえで，プログラマとしての技術的な興味と会社の経営上の課題，すなわち「食べていくための仕事」というのが矛盾することはないのだろうか?「私をはじめ，プログラマはみな自分たちのやりたいことをやっています．私どもは資金もないし，人も少ない．けれども企画をたてて『こういう仕事をやりませんか?』とほかの会社にもちかけることもありますし，逆に『こんな仕事があるのですが』ということで，仕事を依頼されることもあります．ある意味では，恵まれた業界なのでしょうね．食べていくために，無理をして仕事を引き受けることはあまりありません」と酒井氏．

「ただし，受託の仕事も積極的に引き受けるつもりです．新しい技術を仕入れるには，自分たちのやりたい仕事だけやっていてはだめでしょう．技術的に得るものの大きい仕事を選んでやっていきたいですね」．

同社が手がけた仕事のひとつであるRTFJの仕様設計も，もとはある受託業務だったという．最初は「テキストレベルだけでなく，罫線や文字情報も扱える文書ファイルのコンバータを作りたい．ついては変換用の中間ファイルフォーマットを考えてほしい」という内容だった．そこで，社内で独自のファイルフォーマットを作成する一方で，すでにそういったフォーマットが作られていないかどうかを調査したのだそうだ．アメリカにRTFという規格があることを知り，内容をマイクロソフト社に問い合わせたのが縁となってRTFの日本語版の仕様作成へと話が結びついていったのだという．受託業務に対しても積極的に取り組んでいなければ，こうした展開は望めないだろう．

MS-Networksで結ばれた開発環境

ソフトヴィジョンのオフィスは明るく，そして広い．現在の事務所には昨年秋に移転してきたばかりだが，ゆったりしたスペースにパーティションを切って各人の机が配置されているところは，オフィス家具のショールーム並みだ．ただし，打ち合わせをするのにわざわざほかの人の席まで出かけていかなければならない（狭いオフィスのときは椅子の向きを変えるだけでよかった），電話を取り次ぐのに大声を出さなければならなくなった，とぜいたくな悩みはつきないようだ．

プログラマには机と椅子，そしてPC-9800シリーズの32ビットマシンが1人1台確保される．これが各人の開発用マシンとなる．個人用のパソコンは，すべてMS-Networksでサーバにつながっている．サーバには，135MBのハードディスク2台が接続されていて，基本的にはプロジェクト単位でファイルが管理されている．個人用のパソコンにも80

About Company

酒井俊夫社長．1989年1月設立．設立時のメンバー7名は全員がソフトウェアの開発に携わっていたという，いわばプログラマばかりが集まって作った会社．パソコンやワークステーション上のソフトウェア開発，コンサルティング，パッケージソフトの基礎調査，製品企画などを行う．

Location

JR総武線の浅草橋駅（あの秋葉原の次）から徒歩5分．周辺には雛人形や造花を扱う店，問屋などが多く下町情緒も色濃く残る．

Products

日本語入力ユーティリティ
「WinATOK」　9,800円

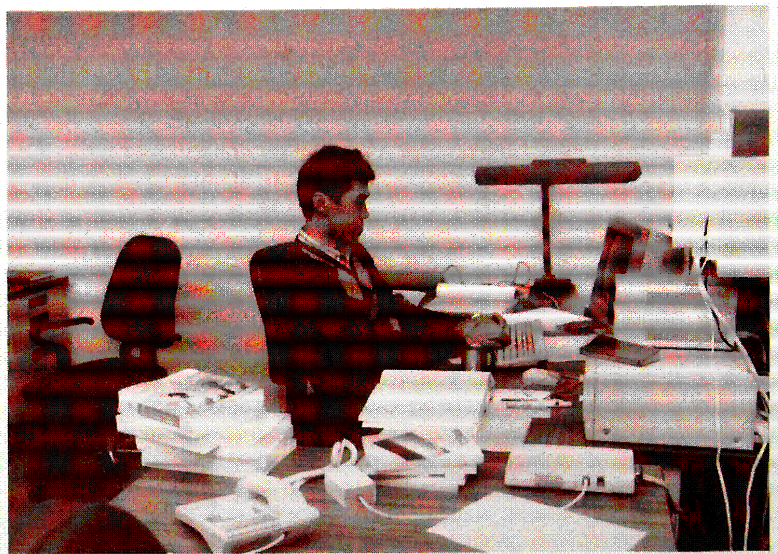
ジャストシステムの日本語フロントエンドプロセッサATOK6，ATOK7をMS-Windows上で使えるようにするユーティリティ．MS-Windows上では，通常Microsoftの定めるカナ漢字インターフェースを備えた日本語FEPしか動作しないため，このインターフェースをもたないATOKは，そのままではMS-Windows上では使えない．WinATOKはMS-DOSのデバイスドライバATOKがWindows上で使えるようなインターフェースを用意する．まもなくWindows3.0対応版も発売予定．販売はアンテナハウス．

PC-9800シリーズ用パソコン通信ソフト
「MILINK」（発売予定）

MIFESライクなエディタを内蔵したパソコン通信ソフト．LISP風の強力なマクロ言語を備え，各種のスクリプトを作成できる．販売はメガソフト．

RTFJ (Rich Text Format Japanese)

RTFJは，日本語ワープロ用の文書ファイルの共通フォーマットである．RTFJのもととなったRTFは，米国のマイクロソフト社が同社のワープロソフトMS-WORDと他のソフトとの文書の互換性を保つために考案したもので，その仕様は一般に公開されている．RTFJはRTFの仕様を日本語用に拡張したフォーマットで，マイクロソフト，アンテナハウスの2社によって共同開発された．現在「一太郎」などの日本語ワープロで作った文書は，通常はそのソフト独自の文書ファイルフォーマットで保存されるため，罫線や文字飾りなどの情報をそのままの形で他のソフトに引き渡すことはできない．他のソフトとのデータ交換にはMS-DOSの標準テキストファイル形式を使うことになる．各社のソフトウェアがそれぞれRTFJの入出力をサポートすれば，罫線や文字サイズなどの情報も含めた形式での文書交換が可能になる．


石野恵一郎氏のデスク．開発プロジェクトの要となる人物だ．

MB~200MBのハードディスクが接続されている．最初はそれぞれ内蔵ディスク40MB，不足分はサーバの容量を増やそうと考えていたが，やはりディスク容量はいくらあっても足りなくなるものらしい．実装されているメモリも，2MB~6MBとマシンによってかなり幅がある．MS-Windows上で動作するアプリケーションの開発担当者は，やはり6MBはないとつらいそうだ．将来的には，UNIXワークステーションをサーバにすることも考えている．これとは別に共用マシンとしてMacintosh，IBM PS/55，各社のAXマシンなどがある．

開発に使うプログラミング言語はCが中心で，一部でアセンブラを使っている．C++を使おうという声もちらほらあがっているが，現状では，大きなプロジェクトで万一人手が足りなくなった場合に，応援を頼みにくいなどの理由からC++をメインにすることはできないという．C言語の処理系は，MS-DOSアプリケーションの場合ならTurbo C，WindowsアプリケーションはMS-Cが標準になっている．

エディタは，各人が好みのものを使っている．MIFES，Vz，Briefが中心だが，さらにこれらのエディタをカスタマイズして他人が絶対さわれないような凝った(?)開発環境を構築している人もいる．実際にプログラムをコンパイルしたり，リンクしたりするときは，サーバからいったん自分のPC上にファイルを持ってきて実行する．

今後の課題は人材確保

プログラマが自ら理想を求めて設立した会社で，自分のやりたい仕事を選ぶことができる．ソフトヴィジョンは，プログラマにとって理想的な環境にも見える．ウィークポイントがあるとすれば，それは人手がないためにやりたい仕事ができないという点だという．確かに10名そこそこの人数では，場合によってはできない仕事というのも出てくるだろう．同社では，今後MS-Windowsのアプリケーション開発にも積極的に力を注いでいきたいというが，まだ知名度が低いために人材が集まらないという悩みがある．酒井社長も，最近では会社の知名度をあげる必要性を感じているそうだ．

●編集子のひとこと

酒井社長の語るプログラマ――単なるコーダーではなく，システムの設計からソフトウェアの完成までをこなせるポジティブな意味でのプログラマは，わが「プログラマーズ・ページ」が描くパソコンプログラマのイメージと相通ずるものがあって感無量．

ワンラインNEWS
ロータスはノート型パソコン専用の表計算ソフト「Loutus 1-2-3/Notebook」を2月下旬発売する．1パッケージで富士通・日本電気・東芝の各製品に対応

information・page

イベント

■3月18日(月)～3月20日(水)
1991年電子ディスプレイ展
国内外の電子ディスプレイデバイス、ディスプレイ応用製品を展示。
会場：科学技術館／入場料：無料
問い合わせ先：日本エレクトロニクスショー協会　TEL 03-3284-1051

■3月22日(金)～3月26日(火)
'91ストア・オートメーション・ショー
最先端の流通情報システム、SAシステムを一堂に展示。
'91カード・ビジネス・フェアも同時開催される。
会場：東京・晴海見本市会場／入場料：1,500円
問い合わせ先：日本経済新聞社事業局総合事業部　TEL 03-3252-8157

■3月26日(火)～3月28日(木)
1991年度精密工学会春季大会学術講演会
講演、パネル形式によるポスターセッションが行われる。
会場：早稲田大学理工学部／参加費：3,000円(会員)、5,000円(非会員)
問い合わせ先：(社)精密工学会　TEL 03-3362-4030

■3月26日(火)～3月29日(金)
1991年電子情報通信学会春季全国大会
一般講演、シンポジウム講演が行われる。
会場：徳島大学常三島キャンパス／参加費：1,000円(会員)、1,700円(非会員)
問い合わせ先：(社)電子情報通信学会　TEL 03-3433-6691

■4月2日(火)～4月5日(金)
コミュニケーションTOKYO '91
電気通信機器・システムの最新技術と機器の展示。
会場：東京・晴海国際見本市会場／入場料：無料
問い合わせ先：通信機械工業会　TEL 03-3231-3156

■4月3日(水)～4月5日(金)
CEPS JAPAN '91
コンピュータ、ワープロによる文書処理統合システムの展示とセミナーが行われる。
会場：池袋サンシャインシティコンベンションセンターTOKYO／入場料：1,000円
問い合わせ先：カーナーズエクスポジションジャパン(株)　TEL 03-3349-8501

■4月10日(水)～4月13日(土)
DISK '91/DISK SYSTEMS '91/DISPLAY '91
磁気、光磁気ディスクメディアやディスク装置、
電子フラットパネルディスプレイなどの製品や製造技術の展示。
会場：晴海・東京国際貿易センター　入場料：1,500円
問い合わせ先：MESAGOジャパン(株)　TEL 03-3359-0894

4月10日(水)～4月12日(金)
スーパーコンピューティングジャパン'91
ハイパフォーマンスコンピュータ、周辺機器の展示とコンファレンスが行われる。
会場：池袋サンシャインシティコンベンションセンターTOKYO／入場料：2,000円
問い合わせ先：スーパーコンピューティングジャパン'91日本事務局　TEL 03-3563-0345

■4月10日(水)～4月13日(土)
Technology Japan '91
FA、LAN、CAD/CAMなどの関連機器を展示。
会場：東京・晴海国際見本市会場　入場料：1,500円
問い合わせ先：日本工業新聞社事業部　TEL 03-3292-3561

セミナー

Microsoft University

■3月26日(火)～3月29日(金)
Microsoft LAN Manager Programming Environment
LANマネージャアプリケーション開発のための入門コース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(4日間)

■4月8日(月)～4月12日(金)
Microsoft LAN Manager Applications Programming
LANマネージャ用アプリケーション開発のための上級コース。
会場：富士ソフトウェアMUセミナールーム
受講料：180,000円(5日間)

■4月1日(月)～4月5日(金)
Microsoft Windows Programming Environment
Windowsアプリケーション開発のための入門コース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(5日間)

Microsoft Universityは、OS/2、Windowsアプリケーション開発者のために定期的に開催されているセミナーです。
問い合わせ先：富士ソフトウェア(株)MU教育センター　TEL 03-3452-7722

NEWS & NEW PRODUCTS

UNIXインターナショナル，1991年版ロードマップを発表

UNIX System Vのベンダー，ユーザーおよび開発者で構成されている団体，UNIXインターナショナル（アジア太平洋本部：TEL 03-5256-6959）は，「UNIX System Vロードマップ1991年版」を発表した．ロードマップは，UNIX System V環境の，将来の発展に向けた戦略的目標や企画，製品開発実施に関する最新スケジュールを表したもので，91年版には，UNIX System Vリリース4を統合したプログラムの出荷予定などがもりこまれている．System Vリリース4（SVR4）は，オペレーティングシステムを超えるものとして，オープンシステムアーキテクチャーを業界に提供する．セキュリティ機能を強化した「SVR4ES」は91年前半，SVR4のシンメトリックなマルチプロセッシングバージョン「SVR4MP」は91年半ば，マルチプロセッシング製品にさらに強化機能を加えた「SVR4ES/MP」は92年に出荷が予定されている．

東芝J-3100にデスクトップタイプ登場．486搭載，EISAバス対応モデルも

（株）東芝（03-3457-2100）は，DynaBookなどで知られる同社のJ-3100シリーズに，デスクトップタイプの新機種，2機種3モデルを発表した．80486を搭載した「J-3100ZX 171モデル」は，32ビットバス対応のEISAスロットなどを採用，VGA対応により高解像度の画像表示が可能．4MBのメモリ，170MBのハードディスクを標準装備し，オプションで本体内に150MBのカセットストリーマを内蔵することができる．80386SXを搭載した「J-3100ZS」は，低価格を実現したフロッピーディスクモデルの「002モデル」，40MBハードディスクを搭載した「041モデル」がある．両機種とも日本語MS-DOS V3.1，ATOK5，ATOK7を標準装備している．また，4月末には日英両用VGAカードが出荷される予定．

■J-3100ZX 171モデル

プロセッサ：80486（33MHz）／メモリ：4MB（最大36MBまで拡張可能）／外部記憶装置：3.5インチFDD×1・170MB HDD×1／拡張スロット：EISAカード専用×3・J-3100専用×2／価格：2,150,000円

■J-3100ZSモデル002／041

プロセッサ：80386SX（20MHz）／メモリ：2MB（最大10MBまで拡張可能）／外部記憶装置：3.5インチFDD×2（モデル002），3.5インチFDD×1・40MB HDD×1／拡張スロット：PC/ATカード用×1・J-3100専用×2／価格：338,000円（モデル002），488,000円（モデル041）

J-3100ZX 171モデル

エプソン，拡張性を重視した32ビットブック型パソコンを発表

セイコーエプソン（株），エプソン販売（株）（エプソンPCインフォメーションセンター：TEL 03-3377-3531）は，同社のPCシリーズの新製品として，拡張性とパフォーマンスを重視した32ビットブック型パソコン「エプソンPC-386BOOK L」を発表した．PC-386BOOK Lは，80386SXを搭載し，PC-286Lシリーズで実績のある拡張スロット（Lスロット）を2基装備，着脱可能なハードディス

ワンラインNEWS
マイクロソフトは「Microsoft Works」など新製品の宣伝にリクルートの運営するFAX情報サービスで効果をあげている．他にもソフト会社やマニア向けも検討している．

NEWS & NEW PRODUCTS

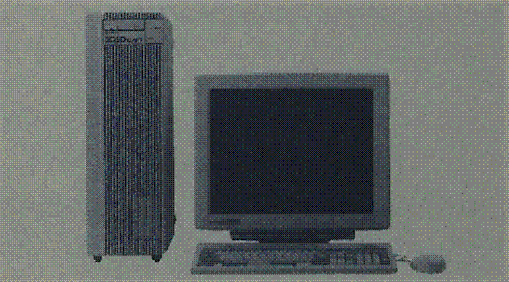
クパック（オプション）を装着すれば，標準装備の2台のフロッピーディスクドライブと合わせて，3ドライブ構成が可能となる．ディスプレイは，8階調の冷陰極管サイドライト方式白液晶を採用，レジューム機能も備えている．

■おもな仕様

プロセッサ：80386SX（16MHz）／メモリ：640KB（最大8.6MBまで拡張可能）／外部記憶装置：3.5インチFDD×2／拡張スロット：Lスロット×2／価格：378,000円，508,000円（20MB HDDパック搭載）

日立，68040搭載のワークステーション2機種を発表

（株）日立製作所（TEL 03-3763-2411）は，同社のエンジニアリングワークステーション「2050Gシリーズ」の上位機種として，CPUに68040（25MHz）を搭載した「2050G/FX」と，高速グラフィックプロセッサを搭載した「2050G/FT」（写真）を発表した．68040は，内部に命令用とデータ用にそれぞれ4MBのキャッシュメモリをもち，処理の高速化をはかっているが，さらにCPUと主メモリ間に256KBのキャッシュメモリを搭載することにより，キャッシュメモリと主メモリ間のアクセス方法を改善し，より高速な処理が可能となった．価格は，7,239,000円（FT），5,870,000円（FX）．



日本電気，N5200シリーズにカラーラップトップなどの新機種を発売

日本電気（株）（TEL 03-3454-1111）は，N5200シリーズに，TFTカラー液晶ディスプレイを採用した32ビットラップトップ「N5200モデル03LC」と，拡張性を重視した普及型ラップトップ「N5200モデル03LS」を発売した．カラー8色表示が可能な「N5200モデル03LC」（1,350,000円）は，同シリーズの最上位機種で，80386SX（16MHz）を搭載し，2つの拡張スロット，3.5インチフロッピーディスクドライブ1台，100MBハードディスクを標準装備している．「モデル03LS」は，2つの拡張スロットを備え，標準で2MBのメモリを搭載，液晶ディスプレイの改良により現行の製品にくらべて，明るさが約10%向上した．3.5インチフロッピーディスク2台を内蔵したFDDモデル（475,000円）と40MB HDDを内蔵したハードディスクモデル（885,000円）がある．

エプソン，200MBの高速大容量SCSI対応ターミナルハードディスクを発売

セイコーエプソン（株），エプソン販売（株）（エプソンインフォメーションセンター：TEL 03-3377-3500）は，SCSIインターフェースに対応した，200MBの大容量ターミナルハードディスク「エプソン HDD-200SC」を発売した．16msecの高速アクセスタイム，32KBのキャッシュメモリの内蔵により，高速なアクセスが可能となっている．価格は，348,000円．

NEWS & NEW PRODUCTS

アイ・オー・データ機器，98セカンドバス規格対応のEMSプラス増設RAMボードを発売

（株）アイ・オー・データ機器（TEL 0762-21-4812）は，同社の提唱した「98セカンドバス」規格に対応した，PC-9800シリーズ用の増設RAMボード「PIO-SB34シリーズ」を発表した．この規格は，サブボードをベースボード上に載せ，ベースボード上のコネクタを介して拡張用スロットバスに接続することにより，1スロット内に2枚のボードをおさめることができる．「PIO-SB34シリーズ」は，サブボードタイプのEMS対応増設RAMボードで，I/Oバンク，EMS，プロテクトメモリ，メインメモリに対応，DRAM素子をファーストページモードで動作させる「ファーストアクセス機構」を備え，ロータリースイッチによる1MB単位のパーテーション設定が可能．価格は，2MBのPIO-SB34-2Mが32,000円，4MBのPIO-SB34-4Mが55,000円．

メモレックス・テレックス，PS/55用Netware対応ハードディスクを発売

メモレックス・テレックス（株）（TEL 03-3273-8905）は，Novell社のLANオペレーティングシステム「Netware」に対応した，日本アイ・ビー・エムのパーソナルシステム（PS/55）シリーズ用SCSI対応のハードディスクシステム「MEMOREX TELEX2102」を発売した．1ドライブ構成でディスク容量1GBの「2102-1モデル」と2ドライブ構成でディスク容量2GBの「2102-2モデル」がある．2102-2モデル1台，または2102-1モデル2台接続時には，「Netware386」の2台のドライブに同じ情報を同時に書き込むミラーリング機能や，複数のドライブを1台と見なすボリュームマルチセグメント機能を利用することができる．価格は，1GBの2102-1モデルが2,280,000円，2GBの2102-2モデルが3,680,000円．

ソフトウェアジャパン，日本語版ピーター・ノートン・コマンダー Ver3.0を発売

（株）ソフトウェアジャパン（TEL 03-3862-3587）は，1991年3月下旬より米国シマンテック社の「ピーター・ノートン・コマンダー Ver3.0」を日本語化し，PC-9801版と東芝J-3100版を同時に発売する．

本製品は，ハードディスクやフロッピーディスクの内容を左右ウィンドウで簡単に素早く見ることができ，左右ウィンドを開いた状態で約100ファイルをウィンドウ内に表示できる．また強力なビュアー搭載により，Lotus 1-2-3やdBASEなどのアプリケーションを起動せずにファイルの内容を調べることができる．価格は，28,000円．

バックス，Windows 3.0に対応したVJE-γを発売

（株）バックス（インフォメーションセンタ：TEL 0427-24-9200）は，MS-Windows 3.0で日本語入力を行うための，日本語入力フロントエンドプロセッサ「VJE-γ Ver.1.0 Windows 3.0対応版」を発売した．半角英数，ローマ字ひらがななど11種類の入力モードを備え，約7万語の基本辞書とユーザ辞書をもち，最大約16万語まで登録することができる．動作には，MS-DOS Ver.3.3以降，MS-Windows 3.0が必要で，PC-9801・RLシリーズ，PC-H98に対応している．価格は，18,000円．

また，同時にPC-9800シリーズ用MS OS/2 Ver.1.21に対応した「VJE-γ Ver.2.2 MS OS/2版」（価格38,000円）も発売した．プレゼンテーションマネージャとフルスクリーンモードの両方で動作する．

ワンラインNEWS
日立製作所は同社のファジィ制御の家電製品に認定マークを制定した．昨年11月に設置した「ファジィ基準策定委員会」で認定を行う．

NEWS & NEW PRODUCTS

三菱商事，SQLBase，SQLWindowsの日本語版リリースを発表

三菱商事（株）は，米国Gupta Technologies社と国内独占販売契約を締結し，共同で日本語化を行っていた「SQLBase」「SQLWindows」日本語版を，1991年第3四半期前半ごろに出荷すると発表した．「SQLBase」は，パソコンLAN上で，クライアントサーバー方式によるデータベースシステムを実現し，従来ミニコンやオフコンで行っていたような情報システムを構築することができる．「SQLWindows」は，Windowsアプリケーションの開発効率を飛躍的に向上させるシステムである．価格は，SQLBaseサーバー日本語版DOS用が459,000円，OS/2用が574,000円，シングルユーザーランタイム版が69,000円．SQLWindows開発システム日本語版（Windows 3.0以上に対応）が298,000円，ランタイム版が35,000円．

アスキー，マルチメディア対応のインフォミックスversion4を発売

（株）アスキー（TEL 03-3797-6506）は，1991年3月末よりリレーショナルデータベース（RDB）「インフォミックスversion4」の製品群の日本語版を発売する．これは，RDBの基本システムである「データベースエンジン」，ネットワーク環境を実現する「ネットワークツール」，アプリケーション開発を行う「フロントエンドツール」の3種類，計8本のソフトウェアから構成されている．従来のインフォミックスに比べて，音声や画像のオンライン・トランザクション処理（OLTP）機能を強化したのが特徴．UNIX上のマルチメディアおよび大規模システムに対応している．

アイ・オー・データ機器など4社，アプリケーションの組み込み規格 MAOIX（マオイクス）を提唱

（株）アイ・オー・データ機器（TEL 0762-21-4812），（株）アスキー（TEL 03-3486-7111），（株）大塚商会（TEL 03-3264-7111），マイクロソフト（株）（TEL 03-3363-1200）の4社は共同で，MS-DOS上で動作するパッケージソフトをハードディスクに組み込む作業を統一した規格「MAOIX（Multiple Application Observer for Install eXecution）」を提唱した．「MAOIX事務局」をアスキー内に設置し，ソフトハウスを対象に「MAOIX」規格の組み込みソフトの提供とサポート業務を行う．

今回の対応機種は日本電気のPC-9800シリーズだが，今後，富士通のFM Rシリーズ，松下電器産業のPanacomシリーズ，日本アイ・ビー・エムのPS/55シリーズなどの機種にも対応する予定．1月末現在，大塚システム研究所，リード・レックスなど11社が協賛している．問い合わせは，MAOIX事務局（TEL 03-3486-4074）まで．

沖電気工業，日本初のフューチャーバス＋採用を発表

沖電気工業（株）（TEL 03-3501-3111）は，米国で開催されたバスコン・ウエスト・ショウで，IEEE896委員会で規格化が行われている「フューチャーバス＋」を，同社のコンピュータのシステムバスとして採用し，同規格の世界的な業界標準化を推進することを発表した．「フューチャーバス＋」は，高いバススループットをもち，マルチプロセッサシステム，フォールトトレラントシステムなどの次世代のコンピュータに要求される機能，デバイステクノロジーやアーキテクチャに依存しない構造と拡張性をもつ，新しいバスアーキテクチャである．同社では，ナショナルセミコンダクター社が開発した「フューチャーバス＋」チップセットを採用する．

Programmer's Program

ヘルプファイルをハイパードキュメント風にリンクさせるユーティリティ
QuickHelp（クイックヘルプシステム）

MS-Cに付属するユーティリティ

「QH」は，Microsoft C Professional Development System Version 6.0（以下MS-C 6.0）に付属するユーティリティで，通常はMS-C 6.0のPWBやコンパイラ，リンカなどから呼び出されてオンラインヘルプとして使われる．マウスでキーワードをクリックすると，リンクされた画面にジャンプしたり，別のテキストを呼び出したりするハイパードキュメント風のユーティリティである．もちろんMS-C 6.0がインストールしてあれば，DOSのコマンドラインから

```
A> QH キーワード
```

とすれば単独でも起動する．HELPMAKEを使えば，ヘルプデータベースをテキストファイルに変換したり，自分で作ったファイルをヘルプデータベースに加えることができる．

ヘルプデータベースを作成する手順

ヘルプデータベースを作成するには，トピックのテキスト，フォーマット，リンク，そして情報を制御するコードをもったソースファイルを作成する必要がある．ソースファイルは，QuickHelpフォーマットと呼ばれるテキスト形式のファイルで，複数のトピックからなる．

トピックは，キーワードのコンテキスト文字列を定義する.contextや，トピックをポップアップさせる.popupなどのドットコマンド，ヘルプを表示するアプリケーションに制御を渡すために使われる制御コマンドとリンクするテキストで構成される．

テキストには一部の色を変更したり，リンク先を示す円記号（¥）に1文字が続くフォーマットフラグを埋め込むことができる．たとえば，テキストの色を変更するには，¥bや¥iを使い，解除は¥pまたは改行を埋め込むことで指定する．円記号そのものを指定するときは，2つの円記号(¥¥)を使う．

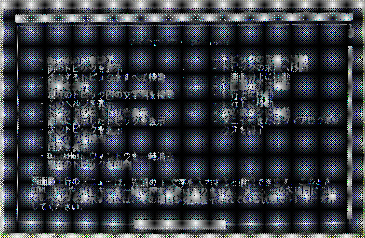
リンクを定義するには，¥aと¥vのフォーマットフラグを使う．スペースではさまれる1つの単語を他のトピックにリンクするときは，単語の後にリンク先のコンテキスト文字列を¥vフラグで囲んで指定する．¥vフラグで囲まれたリンク先は画面に表示されない．また複数の単語など，文章の一部からリンクさせたいときは，ホットスポットの指定を行う．ホットスポットのはじまりを¥aフラグで指定し，終りには単語の場合同様に¥vでリンク先のコンテキスト文字列をはさんで指定する．

ただし，リンク先のコンテキスト文字列やホットスポットを，改行をはさんで続けることはできない．またアットマーク（@）ではじまるコンテキスト文字列は同一データベース内でのみ有効となる．

ほかにもトピックに対して，適当なコンテキスト文字列を定義して，それをリンクのキーワードにすることもできる．そのキーワードが他のトピックに現れる場合，ユーザーがそのキーワードにカーソルを移動してヘルプを要求すると，キーワードに対応するトピックが表示される．この場合は，特別なコーディングは必要ない．キーワードが現れたときはいつでも，結びつけられたテキストへのリンクとして機能する．リンク先は，同じヘルプデータベース中の他のトピック，他のヘルプデータベース中のトピック，通常のテキストファイルが指定できる．

各トピックは，ひとつまたは複数のコンテキスト文字列定義をもち，それによってトピック同士を分離している．また，同じコンテキスト文字列を2か所以上で使用することはできない．

以上のようにソースファイルは，コンテキスト文字列で区切られるトピッ



ワンラインNEWS
コナミ工業は大規模集積回路のCAD用ソフトを開発した．複数の半導体メーカーの設計用言語をサポートし，当初は日本IBMのCADシステムに組込み販売される．

クの集まりから成っている．ひとつまたは複数のコンテキスト文字列が，トピックのラベルとして機能する．ユーザーがデータベース中のキーワードのヘルプを要求するか，リンクを実行すると，どのトピックを表示するかを，コンテキスト文字列が決定する．

Pd ソースファイルは簡単にできる

例としてサンプルを作ってみた（リスト1）．このソースファイルをエンコードするには，DOSのコマンドラインから

A> HELPMAKE /e15 /t /oSAMPLE.HLP SAMPLE.TXT

と入力する．/eはエンコードの指定，数字は圧縮率で，0から15の数字を指定し，大きくすれば圧縮率は高くなる．/tオプションはドットコマンドを使う指定である．以上で拡張子．HLPをもったファイルが生成されるはずである．

さっそく，オリジナルのヘルプデータベースを呼び出してみよう．DOSのコマンドラインから

A> QH -dSAMPLE FORMAT

と入力する．-dは，環境変数HELPFILEで指定されているディレクトリ以外のヘルプデータベースを指定するときに使う．この場合，カレントにSAMPLE.HLPがあれば，画面にFORMATの構文が表示されるはずである．あとはマウスを使って適当なキーワードをクリックするだけである．リンクされているキーワードをクリックすればウィンドウが開く．

ヘルプデータベースのソースは，ドキュメントファイルから簡単に作ることができる．オンラインソフトなど，ヘルプファイルを使って作れば，より便利になるのではないだろうか．

●Microsoft C Professional Development System Version 6.0
販売：マイクロソフト（株）／価格：98,000円／実行環境：286以上のCPUをもつPC-9800シリーズ（ただしハイレゾモードは除く・他に各社各機種対応版あり）・6MB以上の空き容量をもつHDD・MS-DOS Ver.3.1以上もしくはOS/2 Ver.1.1A以上

■リスト1

```
.context sample.txt
.context Format

  FORMAT [[<¥idrive¥p:>] [/¥ioptions¥p]]

.context drive
.popup
フォーマットの対象となるドライブを指定する．/h /e /?オプションを
指定した場合は無効となります．
.context options
  オプション

    {/H|/E|/?}{/6|/9|/M}{{/S|/B}{/V}{/P}}
     全て省略するとメニュー形式で起動する

       ¥a/H : format Hard disk¥v@op08¥v
       ¥a/E : format hard disk Easy¥v@op09¥v
       ¥a/? : Show help¥v@op10¥v
       ¥a/6 : 640K byte format¥v@op03¥v
       ¥a/9 : 9 sector/track format¥v@op04¥v
       ¥a/M : 1M byte format¥v@op05¥v
       ¥a/S : format with System¥v@op01¥v
       ¥a/B : Blank system area¥v@op07¥v
       ¥a/V : format with Volume label¥v@op02¥v
       ¥a/P : don't Press key¥v@op06¥v

.context @op01
.popup
¥iシステムを登録します¥p
FORMAT起動時にカレントドライブのルートディレクトリから
システム（IO.SYS, MSDOS.SYS）とCOMMAND.COMを読み込み，
フォーマットしたディスクに書き込みます．
.context @op02
.popup
¥iボリュームラベルをつけます¥p
フォーマット終了後，ボリュームラベルの入力となります．
全角5文字以内または半角11文字以内で入力します．使用
できる文字はファイル名と同様です．
.context @op03
.popup
¥i2DDフロッピーディスクを640Kバイトでフォーマットします¥p
8セクタ／トラックでフォーマットします．このスイッチは，
ディスクドライブが1MB/640KB両用タイプの場合に有効です．
.context @op04
.popup
¥i2DDフロッピーディスクを720Kバイトでフォーマットします¥p
9セクタ／トラックでフォーマットします．このスイッチは，
ディスクドライブが1MB/640KB両用タイプの場合に有効です．
.context @op05
.popup
¥i2HDフロッピーディスクをフォーマットします¥p
このスイッチはディスクドライブが1MB/640KB両用タイプの
場合に有効です．
.context @op06
.popup
¥iキー入力メッセージを表示しません¥p
.context @op07
.popup
¥iブランクシステムディスクを作成します¥p
システムファイルをあとから追加できるように，約124Kバイトの
空き領域をディスクの先頭に用意して初期化します．
.context @op08
.popup
¥i固定ディスクのフォーマットをおこないます¥p
.context @op09
.popup
¥i固定ディスクのフォーマットをおこないます¥p
/Hオプションより簡単な操作で固定ディスクをフォーマットします．
.context @op10
.popup
¥i簡単なヘルプの表示をおこないます¥p
```

オンラインソフト採集帳 4

ディスクを有効に使うファイル圧縮ユーティリティ LHa Ver. 2.05

「アーカイブ（archive）」を辞書で引いてみると「圧縮」という語は見当たらず「保管」などバックアップという意味で解説されている．バックアップのために複数のファイルを圧縮して保存するユーティリティ，それがアーカイバである．

圧縮されたファイルの拡張子は，アーカイバの種類によって．ZIP，．ARC，．PAKなどが使われている．圧縮ファイルの拡張子に．LZHをもつLHarcは，その高圧縮率で世界的に有名なソフトとなっている．

高圧縮率アーカイバ「LHarc」の上位互換として登場

アーカイバの雄「LHarc」の上位互換性をもった「LHa」がリリースされた．もちろん作者はLHarcでおなじみの吉崎氏である．新しいアルゴリズムの採用で圧縮率や復元速度が改善され，しかもLHarcの上位互換性をもっている．LHarcで圧縮したファイルも復元することができるため拡張子も従来と同じ．LZHである．唯一欠点としてあげられるのは，LHaはメモリの必要量が増大し，大規模なアプリケーションの子プロセスでは実行できない場合があること．

LHA205を実行すると5つのファイルが復元される

LHaは自己復元ファイルとして登録されているので，カレントにLHA205.EXEがあるのを確認してMS-DOSのコマンドラインから

```
A> LHA205
```

と入力する．説明が表示されたのち，復元してもいいか聞いてくるので「Y」と入力すると，カレントに5つのファイルが復元される．このうちLHA_E.BATとLHA_E.BDFは，出力するメッセージを英語にするための差分ファイルとそのパッチファイルである．なお，英語版LHaを作るにはT.TANAKA氏の作成したBDIFFのBUPDATEが必要になる．

これらのファイルを環境変数PATHで指定されるディレクトリにコピーする．RAMディスクやハードディスクなど高速なデバイスが存在するときは，環境変数TMPにそのパス名を指定すれば準備は完了である．

より高速・簡単になったファイルの圧縮・復元は……

LHaでのファイルの圧縮・復元の指定方法はほぼLHarcと同じである．

```
A> LHA A RSDRV RSDRV.SYS ①
A> LHA A RSDRV B:¥*.*……②
A> LHA E RSDRV…………③
A> LHA E RSDRV README.DOC
……………………④
```

HEALTHY?・UNHEALTHY?

④ 食生活の片寄りは仕事にも影響する？

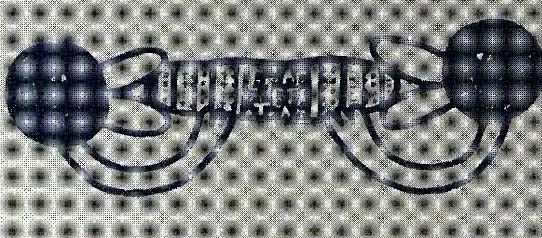


Illustration
Nobuki Fukuhara

忙しいときは食事の時間も惜しんでキーボードにむかっている人も少なくないだろう．そこで，カップヌードル，パンなどで食事をすませてしまうこともある．こうした炭水化物ばかりの食事は，ビタミンなどの不足をまねくことになり，やがては仕事面のマイナスになりかねない．たとえば，ビタミンEやB_1の不足は集中力・注意力の低下をまねくし，B_1，アミノ酸，カルシウムの不足は脳細胞の働きを鈍くする．仕事の内容や生活スタイルによって不足するビタミンの種類も異なるらしい．

ワンラインNEWS
米国のパソコンソフト業界が海外での違法コピーに対し監視の目を強めている．国によっては著作権法の解釈が違い，海賊版が氾濫しているため，

ファイルの圧縮は①のように指定する．この場合，カレントにあるRSDRV.SYSを圧縮して同じくカレントにRSDRV.LZHという圧縮ファイルを作る．②のようにワイルドカードを使った指定も可能で，この場合はBドライブのルートにあるすべてのファイルを圧縮してカレントにRSDRV.LZHという圧縮ファイルを作る．すでにRSDRV.LZHという名前の圧縮ファイルが存在している場合にはそれぞれ圧縮したファイルを追加する．圧縮ファイルを復元するには③のように指定する．この場合カレントにあるRSDRV.LZHに圧縮ファイルをすべてカレントディレクトリに復元する．また④のように復元するファイルを指定することもできる．

```
A> LHA L RSDRV ……………………⑤
A> LHA S RSDRV ……………………⑥
```

圧縮ファイルの内容を確認するL(⑤)，自己復元ファイルを作成するS(⑥)，ほかにもディレクトリごと圧縮されたファイルをそのまま復元するX，圧縮ファイルの更新を行うF，そして圧縮ファイル中の任意のファイルを削除するDなどのコマンドがある．さらに，オプションによって細かい設定ができるようになっている．付属のLHA.DOCにも詳しい説明があるが，MS-DOSのコマンドラインから

```
A> LHA
```

と入力すれば，簡単な使い方も表示される．

LHarcは圧縮率には定評があったが，圧縮・復元時間はほかのアーカイバに比較して遅かった．LHaは，LHarcの上位互換性をもたせた上で復元時間の改善も実現している．この記事は，βバージョンを試用して書いたものだが，本誌が発売されるころには正規版がリリースされているだろう．

■LHa

分類：ファイル圧縮ユーティリティ／登録名：LHA205.EXE（Ver.2.05本体）／著作権者：吉崎栄泰（NIFTY-Serve:SDI00506）／動作環境：MS-DOSが動作するすべての機種／掲載日：1991年1月27日／配布条件：著作権表示を変更しないこと，LHA.DOCも同時に配布すること／備考：一般公開直前のβバージョン／入手日：1991年1月27日／入手場所：NIFTY-Serve

ファイル構成

<LHA205.EXE>

	Name	Size	Date	Time
自己復元ファイル	LHA205.EXE	42275	91-01-27	2:05
復元後のファイル	LHA.EXE	33641	91-01-27	2:05
	LHA.DOC	25671	91-01-27	2:05
	LHA_E.BAT	103	91-01-27	2:05
	LHA_E.BDF	8713	91-01-27	2:05
	HISTORY.DOC	2784	91-01-27	2:05

●ストレスの多い仕事をしていると……C，B_1，E

ストレスをうけると下垂体から副腎に信号が送られ，副腎皮質ホルモンが分泌される．このホルモンが合成されるときにCとEが必要で，ストレス状態が続くとC不足になる．またストレスに打ち勝つ身体をつくるにはB_1．Cはくだものや黄緑野菜，Eは植物油や魚，B_1は豚肉などに含まれる．

●体力の消耗が激しいと……B群とC

エネルギー消費量が多くなると，ビタミン不足をおこして持久力が問題になる．とくにB_1，B_2は消費が激しい．体力の消耗はストレスも生むのでCも必要になる．B_2は牛肉，さば，チーズなどに含まれる．

●タバコの本数が多いときには……C

タバコを1日に1箱吸うと，吸わない人の40倍ものビタミンCが消費されるといわれる．相当量のCの補充が必要．

●お酒を飲む機会が多いと……B_1

アルコールが肝臓で分解されるときにできるアセトアルデヒトという有害物質を処理するにはB_1が必要．酒を飲むと肝臓はたいへんな働きをすることになる．肝臓の負担を軽くするには，B_2，B_{12}などが有効だ．二日酔いの予防にはCを十分にとるといい．B_{12}はカキや肉類に含まれる．

SE-NET拾い読み

「プログラマの内輪話」ボード

本誌オンラインサービス用BBS「SE-NET」は，本誌に掲載したソースプログラムを配布する目的で開設しましたが，4月号よりフロッピーディスクを付録にしたため，ソースプログラムの配布という役割は薄らいできました．今後は，プログラマーズ・ページ編集部と読者，あるいは読者同士のコミュニケーションの場としてご利用いただければと考えています．現在，ご意見などを書き込むために，「DragonPost」「プログラマの内輪話」「雑誌・編集部・ネットへの要望」という3つのボードを用意しています．ここでは，「プログラマの内輪話」からその一部をご紹介します．できるだけログファイルに忠実に掲載いたしますが，編集の都合上一部修正・割愛させていただきました．

■悲惨なプログラマ

グチグチ愚痴っていいでしょか？／Xegnyph．
平成03年01月20日（日）午後10時45分38秒

私なんか，学生時分に行ってたバイト先のソフトハウスの連中にのせられて「ホイホイ」と大学中退してプログラマという，目も当てられない輩です．
コンピュータについての専門教育を受けたわけでも受けるわけでもないので，知識は無いし腕は悪い．そのくせ欲の皮だけは人一倍ツっ張らかっているので扱い難いという，「使えない奴」の典型例です．

貧乏だから休日出勤も残業も人を上回る量をこなさないと喰えないし，そうしていると勉強する暇も気力も無くなってしまうし，だからいつまで経っても悪循環．わかっちゃいるけど抜けられない．

こんな私はいっそこの業界から足を洗うべきなのか…と近頃悩んでいます．
私みたいな悲惨なプログラマーは私の他にいませんか？

Xeg．

RES／Geitassya
平成03年01月22日（火）午前01時55分37秒

ここにも，いるよぉぉぉぉーん何て，言ってたりするが
コンピュータ（汎用）をいじって8年になるけど，勤続年数が増えるにしたがって，私の残業時間が増えてきてますでしゅよ．
これで判った事が1つ，（汎用をいじっていると）1年目，あまり判らないので時間通りに帰れる．
2年目，多少判って来たかな？　と言われ月にして，大体10～20時間位の残業
3年目，これ位は出来るだろうぉぉぉー　と言われ，残業30～40
4年目，さーて！　これ位は，遣ってもらわなくちゃ困る！（会社から）で！　残業50～60時間位
5年目，チーム編制なんか遣ってくれちゃってかなに，中堅どころと言う事で　70～80時間位
6年目，もー　リーダー格として遣ってくれ！　と会社から言われ　残業80～？？　と果てしない時を越え山を越えってな感じ
7年目以降は，色々仕事任されリーダーでなく専門職になって，残業が100～が普通になって来るため，ボーナスなんてこんなものか？　の境地に落ちる（毎月の残業代で，ボーナスに近くなる為有難いなんて思わなくなって，休みくれぇぇぇぇぇぇぇー！　とぼやく）
以上が私の辿った感じですわぁぁぁー
あー　あー　まだ起きていると言う事は，明日は　いいやぁぁぁー
その分，遅くまで仕事してればいっかぁぁぁー
のフレックス時間ではないのに，無理やりフレックスにしてしまう根性まで付いて来てしまうのでしゅぅぅぅぅ

芸達者

実は＊重派遣だったりもして／Xegnyph．
平成03年01月30日（水）午前00時24分58秒

私はS社というベンチャーの零細ソフトハウス（倒産寸前）の社員です．
P社とのコネで紹介されたC社の仲介でY社に派遣され，そのY社の仕事でD社にいます．で，この仕事の都合でN社で作業しています．ウチ（S社）の人間は，私が何処にいるのかまったく把握していません．
ワッハッハッ　　湾岸派遣のほうがいいや…てな気にもなる．
さらに，D社での作業はムチャムチャなスケジュールで押していて，派遣の私が鍵を預かって最後の戸締りをして帰るという…
ワッハッハ　　湾岸派遣のほうがいいや…てな気にもなる．

Xeg．

■ファジーがはやっているようですが

このボードに書くことは…？／Aice
平成03年02月04日（月）午後08時15分55秒

AICEです．こちらには初めての書き込みです．
さて…
「私は，シングルチップマイコンーいわゆる家電製品の中に入っている「マイコン」のプログラミングをしています．家電製品は，日頃から目にしたり使ったりする身近な存在なので，そのプログラム開発となりますと，なかなか楽しいものがあります．」
と嘘八百並べたて，会社見学の学生を騙し続けているのは，何を隠そう私です…
いや，何か違う，今日の書き込みは，こんな事じゃないんだ！！

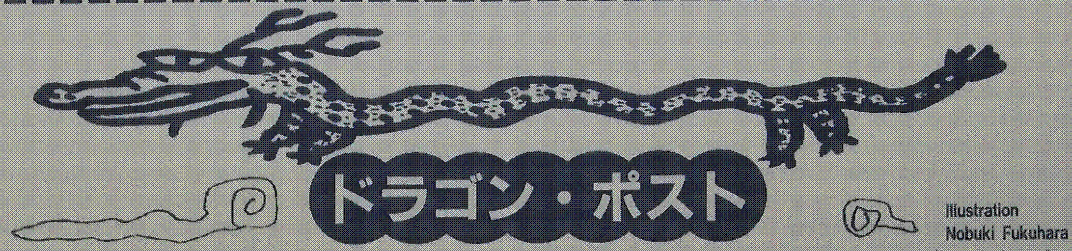
「家電」と言えば（いきなり話はブッとびますが），今，「ファジー」が流行っているようですが，市場を賑わしている「ファジー○○○○」のほとんどが，大嘘，です．
ファジー推論を行うためには，種々の現象ー入力（殆どの場合がA/D）に対して，そのファジー集合（メンバシップ関数）がどのような値をとるのかを他のメンバシップ関数と関係させて導き出さなければなりません．［詳しくは割愛．専門書，読んでね（^_^;)］
しかーし，そんな面倒な演算やら何やらを，内蔵プログラミングＲＯＭサイズ数ｋバイト（最小１ｋ，普通４～８ｋバイト）のマイコンでできるだろうか！！
普通，できる方が不思議ですよね～．
ウチの会社でも，そんなソフトの開発を受けてはいるンですが，演算なんてやりません！（キッパリ）　外部入力から内部のテーブルを引っ張り出してくるだけです．それでも，メーカーの担当者曰く，「これは，ファジー理論を考慮してつくられたデータですので…」
ふーん，数十バイトのファジーデータ，ありがとう．
他もこんなモノだとおもいますけど，今，大々的に宣伝している大部分のファジー製品は，あくまでも
「ファジー理論応用製品」であって，
「ファジー（推論）機能内蔵製品」では，ありません．（一部，例外有り）
「み ん な ， だ ま さ れ る な ！！」
と言いたくても言えない私っていったい…
でも，本当にJAROものですよ，中身知っていると．
と，言ったような事を書き込むとウケるんですか，ここは？
デモ，ゼンブ，ノンフィクションダヨーン
AICE

ファジーなんてあてにしてない／Nagaman
平成03年02月06日（水）午前00時01分50秒

うちのエアコン，ファジー AUTO MODEだと，ぜんぜん温めてくれない．もっぱら，Manual Modeです．掃除機，結局，Manual Modeの最強で使ってる．なんにもいらないから，基本性能に徹した機械が欲しいですね．

ファジィといえば…／Otaku
平成03年02月12日（火）午後04時43分04秒

流行りましたね　一時　AIって　言葉．　ちょっと前にはバイオだのインターフェロンだの．
××洗濯機　××変換　とか　ずいぶん　ありました．
なんだか　懐かしいですね　みんな　どこへ　いっちまったんでしょうか．　ニューロン　そしてファジィのあとは果たして何でしょう．　そういえば　マイコンっていうのも流行りでしたよね（^_^;）　．　いやはや　時の経つのは　腹が立つ．　失礼しました．



ドラゴン・ポスト

Illustration
Nobuki Fukuhara

この欄は，読者からのお便りを中心に構成します．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

★やっぱり雑誌？

やっぱりこのプログラマーズ・ページって雑誌になるのかしらね？　裏表紙には雑誌コードって書いてあるから雑誌ですね．でもね，今まで困っていたのは広告が多くて蔵書するのにカッターで肉を落とす作業が大変!!　指でも切ろうものなら血染めの蔵書よ！　その点，P.P.は良い線いってます．めざせ！そのまま蔵書となるような雑誌を，ちなみに2号は，殿堂（書棚）入りしました．（俊ェ門）

——保存する雑誌は広告をとって書棚へ入れているのですね．雑誌広告も資料として保存する方もいるようですが，いずれにせよ，とっておいていただけるのはありがたいことです．

★C以外の言語にも目を！

私は現在，QuickBASICを使ってあるプログラムを作ろうとしていますが，どの雑誌を見てもC言語ばかりで似たりよったりなので，たまには別の言語に目を向けてもいいのでは？　確かに，C言語がよいことはわかるのですが，いかんせん今までCOBOLやBASICが主になっていたので，これを離れるのはもったいない気がするし……．（AMON）

——おっしゃるとおり，COBOLやBASICの経験があるプログラマは大勢います．QuickBASICのように潜在ユーザーが多い言語に新しい処理系が出てくると，ユーザーはそれまでの経験が生かせるし，その処理系自体が営業的に成功することにもつながります．古くからの言語に新しい環境を取り込んだ処理系の出現が期待されます．

★頭脳労働者としてのプログラマ

2月号の吉田さんに意見があります．コーディングするのはSEと言われてましたけど，SEは主に仕様書やシステム設計で，それを見ながらコーディングするのがプログラマの仕事であり，そのコーディングしたもの（プログラム及びデータ）を入力するのがキーパンチャの仕事ではないでしょうか．プログラマもSE同様立派な頭脳労働者だと思います．（川口　正）

——その通りです．プログラマは創造的な仕事であるという認識は本誌の基本でもあります．またパソコンでは，SEとプログラマを区別するという意識は薄いようです．ハードウェアのほうがパーソナルになっているわけですから……．

★COBOLからパソコンのC言語へ

私は最近，C言語を使う仕事をするようになりました（今までCOBOL）．だから，パソコンというやつも，ほとんど初心者です．MS-DOSも初心者です．私は，主婦もやっているので，のちのちはパソコン買って家でやろうと思っています．だから，COBOLからC言語に移った人でもわかるようなページも作ってください．主婦のパートみたいな仕事させてくれる会社の案内ものせてください．待っています．（のりこ）

——汎用機でCOBOLをやっていて，パソコンの仕事はCという人は多いのでしょうね．そんなあなたのお役にたちたい「プログラマーズ・ページ」です．これからも，よろしくお願いします．

★言語も使い分けの時代に？

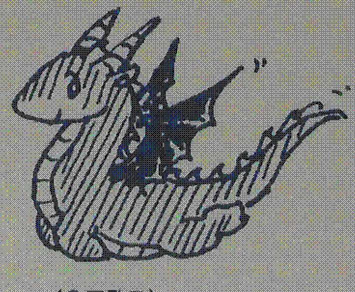
「Cでプログラムを作れば，何でもできる（OSでもアプリケーションでも）」のは事実でしょうが，「定形事務用にはCOBOLかRPG，遊び心を育てないならHyperCardに内蔵のHyperTalk，言語の学習やMacintoshのアプリケーションを作りたいならPascal」という風に，今や言語も使い分けの時代になったと思います．プログラマに求められるのは，プログラムを速く正確に組める能力と，いろいろな言語を効果的に使い分ける能力ではないでしょうか．（高久一伸）

——使い分けはなかなか大変そうですね．パソコンでは，しばらくはCの時代が続きそうだし……．

★ここにもプログラマはいる！

冬の寒い朝，天気がいいと空気中の水蒸気が凍って，ダイヤモンドダストになります．今日は，そのダイヤモンドダストの中を歩いてきました．こんな環境の中にも，プログラマはいるのです．そして，プログラマーズ・ページも発売されているのです．幸せ．（斉藤明好）

——大気の冷たさが感じられるような，気持ちのいいお便りでした．ありがとうございました．



（亀岡啓子）

（敬称略）

バーチャルスペースガイド

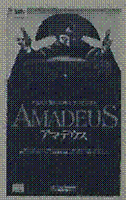
P子 某ソフトハウスで企画営業を担当している．26歳．
C平 某メーカーのシステムソフトウェア開発部門に所属．25歳．

P子 このごろ，レコード売り場にいくとすごいのね．モーツァルトの没後200年とかいうキャンペーンが……．
C平 何年か前はバッハの生誕300年だったし，クラシックは作曲家の生誕何年，没後何年とか，有名な指揮者や演奏家が死んだときしかキャンペーンの張りようがないのかもね．まあ，こういう機会にクラシックファンが増えるのはいいことかもしれない．
P子 なによ，通ぶっちゃって．「アマデウス」でモーツァルトのことおぼえたくせに．その点，私は良家の子女だから，ガキのときから「トルコ行進曲」なんか弾けたんだ．
C平 良家の子女は，ガキなんていわないの！ それに，「アマデウス」だっていいじゃん．映画としてもおもしろいし，おまけにモーツァルトの音楽までたのしめるのだからあんないいものはない．
P子 サリエリがモーツァルトを殺したという仮説．それに軽薄な天才モーツァルトとネクラなサリエリのキャラクターづくりがすごくうまくいっていて，切れのいいドラマになっていると思うわ．当時のオペラの位置とか，宮廷とか……CDや演奏会だけでは絶対にわからないことがわかっちゃうもん．
C平 映画というメディアの力だよね．視覚と聴覚をいっしょにしたという意味で，映画が登場した当時はまったく新しいメディアだったんだ．いま，コンピュータの世界でマルチメディアといっているけど，それが新しい表現の形に結びつくのはまだ先のことかもね．

■アマデウス（1984・米）

監督：ミロス・フォアマン
出演：F・マレイ・エイブラハム

ヨーロッパ文化の中心であったウィーンの都を舞台に，女好きで軽薄ではあるがすばらしい才能をもつモーツァルトと一世を風靡した宮廷音楽家サリエリの確執を描く．努力によって宮廷音楽家の地位を獲得したサリエリは，モーツァルトのたぐいない天才を羨み憎む．全編を通じてモーツァルトの音楽がながれ，「フィガロの結婚」「ドン・ジョバンニ」など，華麗なオペラの舞台もふんだんに織り込まれている．

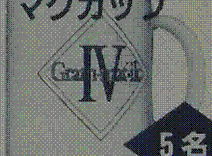


17,650円 ビクター音楽産業／AVC

読者プレゼント

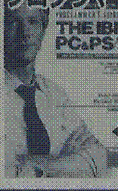
1 LSI C-86

1名
エル・エス・アイジャパン(株)提供

2 Grammatik IV マグカップ


5名
(株)サザンパシフィック提供

応募は本誌に添付のアンケートはがきをご利用ください．なお，応募者多数の場合は，抽選になります．当選者発表は発送をもって代えさせていただきます．

3 THE IBM PC&PS/2 プログラマーズガイド


1名

4 ボーランドOOPツアーTシャツ

3名
(株)ボーランドジャパン提供

付録ディスクのご案内

このフロッピーディスクは5"2HD（8セクタ／トラックMS-DOS）でフォーマットされています．3.5"や2DDなどへのメディアコンバートサービスは行っておりません．なお，オンラインサービスSE-NET（03-5276-1297）でも，同じ内容のファイルを用意しています．

フロッピーディスクには，次のようなプログラムおよびデータを納めてあります．本誌該当記事を熟読した上で利用するようにしてください．なお，ソースファイルのコンパイルなど，別途コンパイラやユーティリティが必要な場合があります．

●「ディスク目次」テキストファイル

ディレクトリ名：ルート（¥）

収録ファイル名

README.DOC

●「特集：C読本」

①Cを知らないあなたのための体で覚えるC言語入門（p.2）

ディレクトリ名：TOKU1

収録ファイル名

LIST01.C　LIST07.C
LIST02.C　LIST08.C
LIST04.C　LIST09.C
LIST05.C　LIST10.C
LIST06.C

②ワークステーションからパソコンへ UNIXのC，MS-DOSのC（p.15）

ディレクトリ名：TOKU2

収録ファイル名

LIST01.C　LIST06.C
LIST02.C　LIST07.C
LIST03.C　LIST10.C
LIST04.C　WHICH.LZH
LIST05.C

③読みやすいプログラムとは？ Cプログラミングのスタイルを決める（p.25）

ディレクトリ名：TOKU3

収録ファイル名

LIST01.C

④資料：C言語関連書籍一覧（p.61）

ディレクトリ名：DATA

収録ファイル名

BOOK.TXT

●MS-DOSツール・ラボ（p.76）

ディレクトリ表示ツール「L.EXE」

ディレクトリ名：DOSLABO

収録ファイル名

CONSOLE.H　L.ASM
DOSFUNC.H　L.EXE
CONSOLE.ASM

●98活用アセンブラ技法（p.98）

サンプルプログラム

ディレクトリ名：98ASM

収録ファイル名

LINERES.ASM　LINETST2.C
LINESUB.ASM　CONDX.INC
RESIDENT.ASM　DOSFILE.INC
LINETEST.C　UTILS.INC

●プログラム「遊」工房（p.70）

「マスターマインドゲーム」

ディレクトリ名：PLAY

収録ファイル名

MILO.C　MILO.EXE

●綴じ込み付録（テキストファイル）

Turboシリーズ　プログラム起動オプション

ディレクトリ名：FUROKU

収録ファイル名

TURBO.TXT

●オンラインソフト採集帳（p.136）

高圧縮書庫管理ユーティリティ「LHa」（著作権者：吉崎栄泰）

複数のファイルをまとめて圧縮したり，逆に復元するユーティリティです．圧縮されたファイルは拡張子が.LZHになります．自己解凍ファイルなので，MS-DOSのコマンドラインからファイル名（LHA205）を入力するだけでいくつかのファイルが解凍されます．

ディレクトリ名：PDSREV

収録ファイル名

LHA205.EXE

●クイズ・ランダムスクリーン（p.143）

クイズ解読ツール「PAZLE」

本誌掲載のクイズランダムスクリーン（p.143）の画像をディスプレイ上に再現し，マウスを使って並び替えることができます．

ディレクトリ名：RANDOM

収録ファイル名

RNDSCREN.R1　RNDSCREN.RGB
RNDSCREN.G1　PAZLE.EXE
RNDSCREN.B1　PAZLE.DOC
RNDSCREN.E1

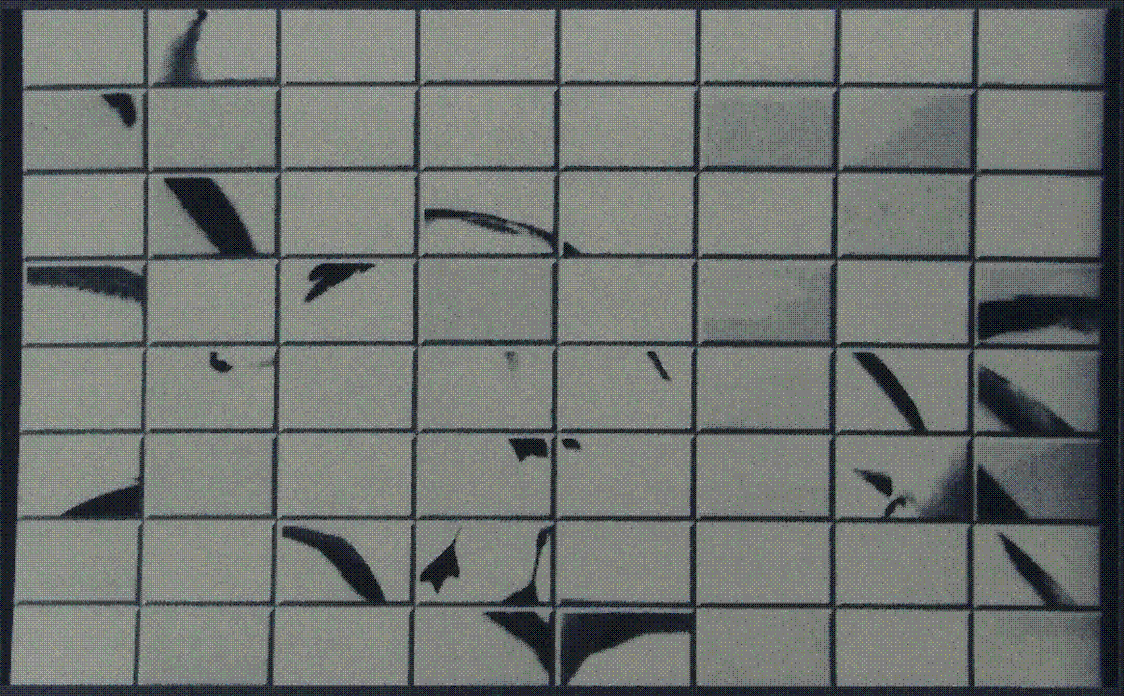
＊使用上の注意

本誌掲載プログラムおよびデータを利用するにはMS-DOSの基礎知識が必要です．また，このディスクにはMS-DOSのシステムは入っていません．別途MS-DOSが必要です．使うときには，必ずREADME.DOCを読んでください．

クイズ・ランダムスクリーン

ある１枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：「いつも群れ飛ぶ○○○さえ……」



正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．

応募方法：本誌に添付のアンケートはがきをご利用ください．

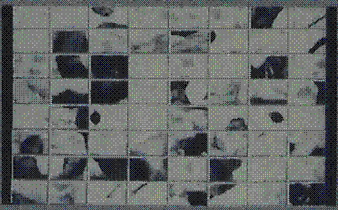
締め切り：1991年４月17日（消印有効）

なお，当選者の発表は発送をもって代えさせていただきます．

＊画像データは，BBSでもサービスしています．

3月号の問題と正解

【問題】



【正解：猫】

月刊プログラマーズ・ページ オンラインサービス（SE-NET）のお知らせ

付録にフロッピーディスクをつけたため，ディスクによるプログラムサービスは中止いたします．オンラインサービスのBBSは，多数の読者の登録をいただいておりますので，今後も継続いたします．編集部とのコミュニケーションの場としてご利用ください．

●BBSへの接続

SE-NET　03-5276-1297

①接続するとIDの入力を求めてきますので，GUESTと入力しリターンキーを押してください．
②お知らせを表示後，メインメニューのコマンドプロンプトが表示されます．
③月刊プログラマーズ・ページへのご意見などは「ドラゴンポスト」ボードに書き込んでください．
④ファイルの転送が終了したら，Oと入力して回線を切ります．

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ5月号

（4月18日発売）

付録：5″ 2HDフロッピーディスク付き!!

特集

基礎力としてのアセンブラ

実行速度の速いプログラムを書くために……
よりコンパクトなプログラムが必要だから……
ハードウェアに近い制御を求められるときに……
アセンブラはソフトウェア開発に欠かせない存在である．
また，CPUと密接な関係をもつ記述言語であるだけに，
ハードウェアをより深く理解する手段ともなる．
次号では，「いまどきのアセンブラ」を徹底解説する．

月刊プログラマーズ・ページ　1991年4月号
1991年4月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘＋勝山まゆみ
デザインスタッフ　佐々木幹夫＋SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋石崎健治
協力　（株）ドモン・マインズ＋シグマCC（株）＋根本裕三郎
＋佐藤裕美＋小林弘人（図版デザイン）＋本田憲章
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1991 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます．
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください．

EDITOR'S PAGE

●4月号から本誌の体裁が少しかわった．フロッピーディスクサービスへの要望が強かったことに応えた形だが，それに伴って，本文のほうも構成を変更した．カラーページがなくなるなどご不満の向きもあるかと思うが，内容的にはプログラミングの新連載を設けるなど，アマ・プロを問わずプログラマを対象にした雑誌として充実を心がけたつもりだ．今後とも読者のご支援をお願いしたい．（TA）

●そろそろ各社から日本語Windowsが出そろってきたようだ．いつものことだが，アメリカでWindows3.0が発表になってからほぼ1年遅れの日本デビューである．そのアメリカでは，ついにBorlandがWindows対応のBorland C++を発表した．Windows SDKを不用にしたこの処理系の登場で，アメリカのWindows市場はさらに活気づくに違いない．Windows用FreeSoftwareも数多く登場してくることだろう．（嘉平）

●今月号のクイズ・ランダムスクリーンの問題の元になった写真を撮影したのはなにを隠そう私です．ヒントを読んだ人は当然答えは〇〇〇だと思うでしょう？　ここだけの話ですが，あれは××××なのです．識別の場合のポイントは目の周辺のリングと足の色なのですが，モノクロ写真では所詮無理だろうなぁ……付録ディスクには画像データも入っています．パズルにチャレンジして私の傑作（?）を堪能してください．（kos）

●先日，自転車を買った．それだけで休日の過ごし方，行動範囲が一変した．なにしろ，自転車に乗るのは中学生以来，「もしかして，乗れなかったらどうしよう」と心配だったが，ちゃんと乗れた．記憶喪失になっても，自転車乗りと泳ぎだけは忘れないらしい．視点が変わると，歩いているときには気づかなかった東京の道を再認識する．歩道は放置自転車，看板などで溢れかえり，気分よくサイクリングするには危険すぎる．（ま）

●愛読者カードを拝読すると結構目につくのが，「C言語ばかりに片寄らないでほしい」というご意見．今回の特集，一部にはご不満の方もいらっしゃると思う．しかし言語というのも，マスコミにおける松田聖子と同じで，好きだ嫌いだのと騒がれているうちが花．本当にその寿命が尽きるのは，嫌いだとさえいう人がいなくなったとき．皆様のこだわりのあるご意見をお待ちしてます．（K）

●今月号からディスクを付録にするとかで，なかなか太っ腹である．他の雑誌の付録のディスクの名称を眺めながら，名前をいろいろ考えてみた．目の肥えた読者に対して「豪華」とか「特別」という言葉ではきっとインパクトがない．某雑誌の広告をパラパラ捲って，ついにすばらしい案が浮かんだ．それは「『一流ブランド』オリジナルフロッピーディスク」である，もちろん却下だった．僕にはコピーライターの素質が全くないらしい．（よ）

お詫びと訂正　本誌3月号で紹介した「Mind5.1」の問い合わせ先がまちがっていました．正しくは「㈱マイクロソフトウェアアソシエイツ 03-3486-1411」です．訂正するとともに，ご迷惑をかけた方々にお詫びいたします．

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊プログラマーズ・ページ READER'S SURVERY CARD

愛読者カード

本誌の号数　　年　　月号

本誌を知った理由
□1. 書店店頭　□2.他社出版物(　　　　)
□3.小社出版物　□4. DM　□5.口コミ

パソコンの所有形態(機種名)
□1.個人所有　(　　　　)
□2.職場にある(　　　　)
□3.持っていない

本誌を購入した書店　　市(区)　　書店

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事(いくつでも)

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト3
パソコン雑誌 ①　②　③
ソフトウェア ①　②　③
パソコン本体 ①　②　③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください(「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください)。

月刊プログラマーズ・ページ 広告資料請求カード

8	会社(学校)名		9	所属(学部)		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業
④建設業　⑤通信・運輸　⑥システム・ソフト
⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ(ハード)技術者　②コンピュータ(ソフト)技術者　③エレクトロニクス技術者
④プログラマ　⑤その他の技術者　⑥専門職
⑦事務・営業職　⑧経営管理職　⑨商工サービス
⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　年　　月号

希望項目 ＼ 広告資料請求No.												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

定期購読のご案内

綴じ込みの振替用紙を使って、本誌の定期購読をお申し込みになれます。
1年間12冊=11,760円(本体11,417円/税343円)
郵送料サービス

郵便はがき

料金受取人払

麹町局承認

3908

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ
READER'S SURVEY CARD 愛読者カード

氏名	性別 □1男 □2女	生年月日 19 年 月 日
住所 〒		電話 ( ) ー

職業			ユーザー区分
□1ソフト技術者	□2ハード技術者	□3 技術系会社員	□1 プロ
□4事務系会社員	□5営業系会社員	□6教師	□2 マニア
□7一般公務員	□8経営管理職	□9理系学生	□3 一般ユーザー
□10文系学生	□11その他( )		

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払

麹町局承認

3910

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係 行

＊当社で記入

1	SE	2	受付No.	

月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓 名	5	生年月日	19 年 月 日

6	自宅住所	〒 電話 ( )	資料送付希望先の番号に○印をつけてください。
7	会社住所	〒 電話 ( )	

払込票
通常払込料金加入者負担
口座番号 東京 4 154493
加入者名 株式会社 翔泳社
金額 11760
※ 払込人住所氏名
受付局日附印
備考
切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。
払込通知票
通常払込料金加入者負担
口座番号 東京 4 154493
加入者名 株式会社 翔泳社
金額 11760
料金 払込み 特殊 円
備考
受付局日附印
払込人住所氏名 ※（郵便番号　　） TEL（　　）
この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）
各票の※印欄は、払込人において記載してください。

名度の高い、米国ボーランド イ
]発したプログラミング言語
ｐでも『Turbo C』は、優れ
コンパイルスピードを誇り、ソ
も強い支持を得ている。また
」は、その使い勝手や機能・価
され、1990年には日経PB社主
ぶパソコン・ベスト・ソフト」の

たのが、マイクロソフトウェア
）である。MSAは、1980年の
フトウェア製品を、日本語化し
-ランド インターナショナル社
関係を維持しており、同社との
ャパンを100%出資で設立する
ている。編集部はMSAに、企

…したり、全体のプロジェクトの
…マネージメントする。
…他営業、総務、業務等を加える
…在、MSAの社員総数は約70名
…。

り上げている。MSAの現場は11年目の現在、ボーランド製品を中心に、さらにUNIX関連ソフトウェア開発、受託業務開発とその開発内容を拡大しつつある。

メインとなるボーランド製品では、TURBOシリーズを初めとするソフトウェア群を日本語化し、国産コンピュータに移植している。MSAの社員は同じフロアにある（株）ボーランドジャパンに出向するという形をとる。この子会社設立は、「BORLAND」の

受託業務チームは、主に金融関係のアプリケーションの中で、パソコンやワークステーション側でのマン・マシンインターフェース部分の開発を行なっている。

これらの開発3部門に加え、MSAにはさらに企画部門も設けられている。企画部門では、「テクニカルサポート」と「プロダクトマネジメント」が主体で、サポートはプログラマが担当している。マネジメントとは、次に移植する場合の仕様を決めたり、移植の機能

篠原 幸吾、入社7年、33才。MSAがまだ設立まもない頃に入社、現在、取締役企画部長。

定期購読のご案内

■定期購読は、1年間12冊と限定させていただきます。

■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。

■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。

■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。

■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

この欄は、加入者あての通信にお使いください。

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係 行

2	受付No.	

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名	5	生年月日	19　年　月　日

綴じ込みの1

□10文系学生　□11その他(　　　　　　)　ユーザー

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　電話（　　）
7	会社住所	〒　電話（　　）

資料送付希望先の番号に○印をつけてください。

ちょっと気になるこの会社

株式会社マイクロソフトウェアアソシエイツ

プログラマーの間では知名度の高い、米国ボーランド インターナショナル社が開発したプログラミング言語『TURBOシリーズ』。中でも『Turbo C』は、優れた総合開発環境と超高速のコンパイルスピードを誇り、ソフトウェア開発の現場でも強い支持を得ている。また『TURBO PASCAL』は、その使い勝手や機能・価格等がユーザーに高く評価され、1990年には日経PB社主催「第3回 23万読者が選ぶパソコン・ベスト・ソフト」のベスト評価賞を受賞した。

このブーミングを仕掛けたのが、マイクロソフトウェアアソシエイツ（略称MSA）である。MSAは、1980年の創立以来、海外の優秀なソフトウェア製品を、日本語化して市場へ導入してきた。ボーランド インターナショナル社とは、7年に渡るビジネス関係を維持しており、同社との合意で(株)ボーランド ジャパンを100%出資で設立するなど、着実に業績を伸ばしている。編集部はMSAに、企画部長・篠原氏を訪ねた。

MSAの業務セクション

アメリカで『turbo pascal』が発売された当時、MSAの岡田社長はその最初の広告に着目して自らボーランド社に赴き、代理店契約を交わした。その先見の明が、今日の発展の基盤を造り上げている。MSAの現場は11年目の現在、ボーランド製品を中心に、さらにUNIX関連ソフトウェア開発、受託業務開発とその開発内容を拡大しつつある。

メインとなるボーランド製品では、TURBOシリーズを初めとするソフトウェア群を日本語化し、国産コンピュータに移植している。MSAの社員は同じフロアにある(株)ボーランドジャパンに出向するという形をとる。この子会社設立は、「BORLAND」のネームで製品を認知させたいという、ボーランド社の戦略に基づいている。

UNIX関連のチームは、オペレーティングシステム上の各種ツールの開発に加え、オリジナルのスプレッドシートソフト開発や、ユーザーからの要求に応じたカスタマイジングなどを行っている。

受託業務チームは、主に金融関係のアプリケーションの中で、パソコンやワークステーション側でのマン・マシンインターフェース部分の開発を行なっている。

これらの開発3部門に加え、MSAにはさらに企画部門も設けられている。企画部門では、「テクニカルサポート」と「プロダクトマネジメント」が主体で、サポートはプログラマが担当している。マネジメントとは、次に移植する場合の仕様を決めたり、移植の機能を定義したり、全体のプロジェクトの進行をマネージメントする。

その他営業、総務、業務等を加えると、現在、MSAの社員総数は約70名である。

篠原 幸吾、入社7年、33才。MSAがまだ設立まもない頃に入社、現在、取締役企画部長。

指示は無いものと思え

MSAを構成する70名のスタッフは、どのようにして入社してきたのだろうか。

「人はどのセクションでも足りないんですが、必要としている人材がなかなか確保できないのが現状です」と篠原氏。時々、ボーランドやTURBOシリーズのファンとおぼしき人が、面接に来ることもある。基本的にはC言語・アセンブラに精通していることが条件であるが、実際にMSAが手掛けている開発内容の水準に見合う人材は、なかなか見つからないそうだ。どうしても高い技術を持つ人材が必要なため、ほとんどが経験者採用である。入社後は、いきなり少人数編成のチームに入って開発に取り組むことになる。C言語の普通のプログラムができる程度ではだめで、PC-9800やIBM PCのハードの詳細を把握した上で、プログラミングしていかなくてはならない。

MSAは、命令されたことしかできない人や、企業の歯車になってしまうようなタイプの人間は必要としていない。レベルの違いこそあれ、あるセグメントで任された仕事はその担当者が決めればいいという方針で動いている。

「与えられたレベルの中では自分が責任者。ほとんど指示は無いものと思え」が、トップの声である。何年か経ったら開発テーマを自分で持てるまでにならないと、マシンに向かってもすることがなくなりそうだ。指示待ち族ではかなり居づらい。

男女の区別は一切ないが、技術系の女性はたった3人である。

「たまたまムサ苦しくなっちゃてるだけなんです。女性にもっとたくさん入っていただきたいですね、ほんとは。TURBOシリーズを移植しているチームなんかは〝変わり者〟っていうか、全員がTK-80時代からのパソコンユーザーなんていう、とんでもない人間の集まりですからね。ムサ苦しさを無くしたいですよ」と、男性陣の本音がこぼれた。

徹底的に自由

「指示は無いものと思え」とは、社員にとっては重たい台詞だ。社内には常に緊張感がみなぎっているように思える。が、篠原氏の答えは意外だった。

「求人向けの会社案内を作るんで、社員の声を集めたときビックリしたんですけど、もうみんなが口を揃えて〝やたら自由だ〟って言うんですよ」

そもそも岡田社長の信念が〝自由〟だった。自発的に仕事をする人間が集まった組織をつくりたい、だから社名にも「アソシエイツ（仲間、同僚の意）」と付けた。ガチガチの管理体制を徹底的に嫌った社長は、まさに自由な企業を実現したのだ。〝指示は無い。自由にやれ〟は、ソフトウェア開発に対して確固とした夢を持つ技術者にとっては、夢を形に出来る可能性を幾重にも広げる環境を意味する。

社員には必ず2～5、6台のパソコンが行き渡っている。UNIX系あるいはパソコン系のネットウェアのサーバーが、社内のいたるところに配置され、それぞれがネットワークで繋がっている。どこからでも双方にアクセス出来、ファイルの共有やメールの交換が自在だ。UNIXのワークステーションは10台以上は置かれている。

「会社だといい環境でコンピュータが触れるから、いわゆるハッカー達は休日でも出勤してしまうほど」らしいが、つられて休日出勤せざるを得ないような環境だということか。

「いや、もちろん休みたいと思う人の方が多いですよ」の答えに安心した。

『リフレッシュ休暇制度』を設けているのが、MSAの特徴と言えよう。入社してから偶数（2、4、6…）年目には、有給休暇と別枠で連続2週間の休みが取れる。休日の消化率は高い。

また、忘年会にライブハウスを借り切って、自分達のバンド演奏で盛り上げるところも、自由で独創的なMSAならではのパフォーマンスだ。

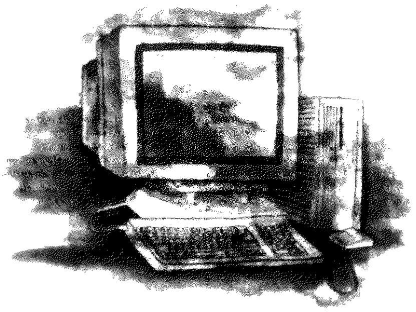
ソフトウェア開発は、まさに〝創造性〟が要求される仕事だ。人と同じことや誰にでも出来ることで満足していては、革新的なものは到底つくれない。企業の歯車ではなく、自らが原動力となるモーター的人材を揃えたMSAは、ボーランド インターナショナル社と成長を共にしながら、「夢のソフトウェア」を形にしてゆく企業である。

『この心の叫びを聞いてくれ！』のコーナーを新設します。

このコーナーでは皆様の仕事、職場環境などに対する不満、お怒りを募集しています。
官製ハガキにて下記にお送りください。抽選で毎月10名に粗品を進呈いたします。
東京都千代田区平河町2-4-14　平河町KSビル
〒102　㈱翔泳社　マーケティング部　TK係

ELTEX

常にパーフェクトでありたい。



ユーザーや上司が求めているものを機械的に作り上げていく。そんな顧客志向型が、本当のＳＥ・プログラマーと呼べるであろうか。常に思考し続け、より高い価値観を徹底追求し、期待以上のものを創造していく。それが私たちの本当の姿なのだ。提案型ソフトウェアハウスのエルテックスは、そんなＳＥ・プログラマーたちの集まり。たえずパーフェクトなものを求め続けていく集団なのです。

●

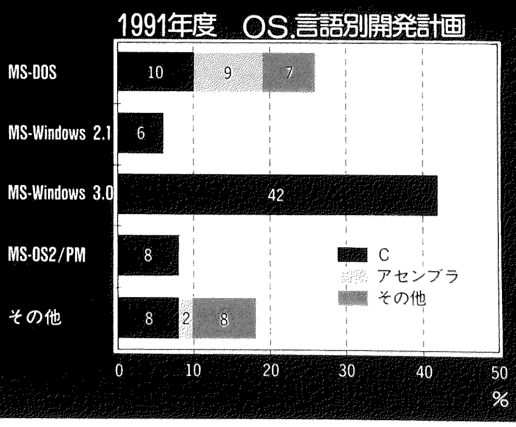
人材募集

会　社　概　要

設立	昭和60年12月14日
資本金	1,000万円
売上高	2.3億円（平成１年度）
事業内容	パソコンを中心としたシステムの提案・開発・販売 ①CD-ROM・LDなど新しいメディアに対応したシステムの開発 ②ネットワークの端末システムの開発 ③特定業種向けCADの開発・販売 ④基本ソフトの開発
特徴	大手クライアント向けのシステム開発が多く、ネットワーク関連システムやマルチメディアなど幅広く対応している また、CADを中心とした自社プロダクツにも積極的で、その比率も増加している 基本ソフトの分野では、ハードウェア専門のベンチャー企業とジョイントし、OEM製品のBIOS・ユーティリティなどを企画から手掛けている
取引先	三菱電機、三菱自動車、日本航空、旭硝子ほか
関連会社	株式会社ソーテック、学校法人福井情報ビジネス総合学院

募　集　要　項

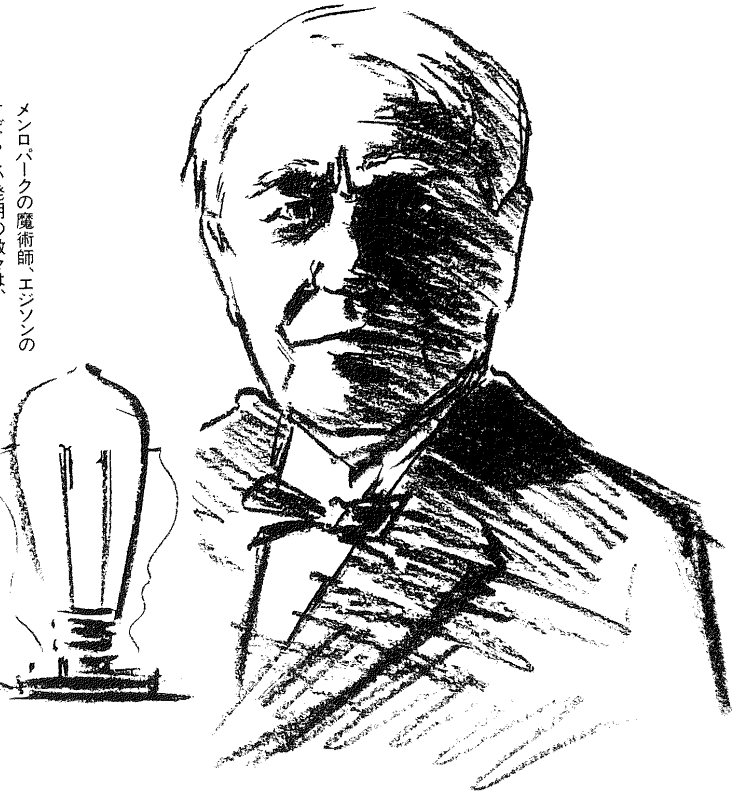
職種	①システムエンジニア　②プログラマー
資格	専門卒以上20〜36歳の方 C言語・アセンブラ言語経験者大歓迎 未経験でも好奇心・向学心の旺盛な方は歓迎します
給与	17万円（専門卒初任給）〜46万円（36歳課長、諸手当含む） 24歳例：23万4千円（情報処理１種手当・住宅手当含む） 30歳例：31万6千円（中級SE、情報処理１種・住宅手当含む）
手当	時間外手当 役職手当（5千円〜6万円） 住宅手当（1万5千円） 資格手当（2種：5千円〜特種：2万円） 通勤手当
賞与	賞与年2回（平成１年度実績年間5か月） 決算賞与年１回（平成１年度実績１か月）
勤務地	本社（横浜駅近く） 派遣・出向や社外での開発作業はありません
勤務時間	フレックスタイム制（コアタイム10：00〜16：00） 試用期間中は、標準時間9：00〜17：45
休日休暇	完全週休2日制（土日）、祝日、年3回連休（GW、夏期、年末年始）、有給休暇、特別休暇
福利厚生	社会保険完備、退職金制度、財形貯蓄、社員旅行、海外研修、定期健康診断、社員寮（制限有）、スポーツクラブ法人会員
応募方法	電話連絡の上、履歴書（写真貼付）、職務経歴書を持参し来社してください　郵送も受け付けます
連絡先	〒220　横浜市西区南幸二丁目20番5号　東伸24ビル5階 Tel(045)322-9661　総務課



まずは、本文連載記事「**プログラマの仕事場**」をごらんください。私たちの会社や業務の内容がよく分かると思います。取材のときにもお話ししたのですが、より良い仕事をするために、より多くの仲間が必要になってきました。私たちの仲間に加わりませんか。自分たちの会社、自分たちの仕事として、共に苦労し、共に喜んでいただける方であれば、プロジェクトマネージャやテクニカルライタも求めています。会社の雰囲気や開発環境、仕事の内容などについては本文の連載記事で紹介していただけると思いますので、ここでは待遇面などについて述べさせてもらいます。報酬は原則として年収ベースで考えます。一昨年、会社設立の年は安月給でした。昨年はソフトウエア業界としての世間並みにはなりました。今年は人材募集をしなければならない程度(?)です。十年後は、…? あなた次第です。福利厚生? 施設は何もありません。何を作りましょうか? 何が欲しいですか? 飲み会、旅行などの補助は随時行っています。社長は冬になると落ち着かなくなりますが、一カ所に別荘を作るよりは毎回違う場所に行きたいと思っています。住宅手当? 一応ありますけど、今のご時世一万や二万の住宅手当があってもねぇー…。要は、報酬の総額です。こんな具合ですので、まだまだこれからの会社です。将来性だけはどこにも負けないつもりですが、それも皆さん次第です。是非仲間に加わっていただいて、すばらしい会社に育てる力になってください。最後に、将来に対する夢物語を少しだけ聞いてください。「プロの技術に裏付けされた、真に利用者の立場に立ったソフトウエアを」というのが設立時からの理想です。ソフトウエアの開発専門会社として「さすがにソフトヴィジョンが開発したソフトだ」と言われるように。また、アメリカに負けない日本独自のオリジナリティあるソフトウエアを世界に向けて供給できるように。そんな日が来ることを夢見つつ少しずつ歩んで行くつもりです。

プログラマを求めています。

メンロパークの魔術師、エジソンのすばらしい発明の数々は、
時代が何を必要としているのかを察し、
人々の暮らしのために生まれてきたものばかりだ。
コンピュータのためのプログラムではなく、
利用する人のためのプログラムを。
ソフトヴィジョンはいつもそう考えている。



■募集要項

職　　種	プログラマ(Windows／MS-DOS／UNIX) プロジェクトマネージャ　技術営業 テクニカルライター
資　　格	大卒以上または実務経験2年以上 20～35歳くらいまでの方
給　　与	経験・能力を考慮の上，当社規定により優遇 例：大卒22歳　19万円　30歳　30万円
待　　遇	昇給年1回(4月) 賞与年2回，業績により決算賞与 (90年実績：成績により8～14ヵ月)
休日休暇	完全週休2日制，祝日，年末年始，有給休暇 (初年度15日，連続休暇有)
勤務時間	9:00～17:30　フレクスタイム有
応　　募	お気軽にお問い合せのうえ，とにかく一度様子を見においでください。応募の秘密は厳守します。その後，履歴書，職務経歴書をご用意願います。　（担当　酒井）

■会社概要

設　　立	昭和64年1月5日
資 本 金	1,750万円
従 業 員	8名
年　　商	1億2000万円
代 表 者	取締役社長　酒井俊夫
事業内容	①コンピュータソフトウェアの企画・設計・製作 ②コンピュータ関連の調査・研究 ③コンピュータ関連のコンサルテーション

Soft Vision
株式会社 ソフトヴィジョン
〒111 東京都台東区浅草橋3-24-8 伊藤ビル
TEL03(5687)8571　FAX03(5687)8718

Turboシリーズ プログラム起動オプション

Turbo C++　Version 1.01

●TCC.EXE（C++コンパイラ）構文：TCC　[options]　file[s]

〈options〉	後ろにハイフン（-）をつけると，そのコマンドをキャンセルすることができる
-1	80186拡張インストラクションコードの生成
-2	80286プロテクトモードインストラクションコードの生成
-A*x*	*x*:なし　ANSIキーワードのみを使用する
	T　TURBO C++のキーワードを使用する
	K　Kernighan and Ritchieのキーワードのみを使用する
	U　UNIXキーワードのみを使用する
-B	インラインアセンブリ処理のためにアセンブラを呼び出す
-C	コメントのネストを許可
-D*xxx*	マクロの定義
-E*xxx*	使用アセンブラの設定
-G	実行速度優先の最適化（-G-　サイズ優先の最適化）
-I*xxx*	インクルードファイルのディレクトリ設定
-K	char型のデフォルトをすべて符号なしに指定
-L*xxx*	ライブラリファイルのディレクトリ設定
-M	リンカにマップファイルの生成を指示
-N	スタックオーバーフローをチェックする
-O	ジャンプ命令の最適化
-P	オブジェクト指向プログラムの許可
-Q*xxx*	*xxx*:e　使用可能なEMSメモリをすべて使う
	e-　EMSメモリを使わない
	x　使用可能なプロテクトモード用拡張メモリをすべて使う
	x-　拡張メモリを使わない
	x=*n*　他のプログラムために*n*キロバイト残して，残りを使う
-S	アセンブラコードの出力
-T*xxx*	アセンブラオプションの設定
-U*xxx*	マクロ定義の解除
-V*x*	仮想テーブルの制御
	x:なし　拡張.OBJ形式を使う（既定値）
	s　標準.OBJ形式のみを使う
	0　拡張・標準.OBJ形式を組み合せて使う
	1　拡張・標準.OBJ形式を組み合せて使う
-X	コンパイラの自動依存情報の出力を無効にする
-Y	オーバーレイコードの生成を有効にする
-Yo	コンパイルされたファイルをオーバーレイにする
-Z	レジスタ操作の最適化
-a	ワードアラインメント（-a-　バイトアラインメント）
-b	enums型のサイズをワードにする（既定値）
-c	コンパイル後，リンクを行わない
-d	重複する文字列を連結する
-e*xxx*	生成する実行ファイル名の設定
-f	浮動小数点演算をエミュレートする（既定値）
-ff	高速浮動小数点演算を行う（既定値）
-g*n*	*n*個の警告・エラーでコンパイルの中止
-i*n*	識別子の文字長を*n*個に設定
-j*n*	*n*個のエラーでコンパイルの中止
-k	標準スタックフレームの作成（既定値）
-l*x*	リンカオプションとして*x*を渡す
-m*xx*	メモリモデルの指定
	xx:c　コンパクトメモリモデル
	h　ヒュージメモリモデル
	l　ラージメモリモデル
	m　ミディアムメモリモデル
	m!　ミディアムメモリモデル，DS!=SSを想定する
	s　スモールメモリモデル（既定値）
	s!　スモールメモリモデル，DS!=SSを想定する
	t　タイニーメモリモデル
	t!　タイニーメモリモデル，DS!=SSを想定する
-n*xxx*	出力ファイルのディレクトリ設定
-o*xxx*	オブジェクトファイル名の設定
-p	「パスカル引数受渡し手順」で関数の呼び出しをする
-r	レジスタ変数使用の許可（既定値）
-u	下線記号の付加をオンにする（既定値）
-v	ソースレベルデバッグ情報の付加
-w*xxx*	警告の出力を抑制
-y	行番号情報を付加
-zA*xxx*	コードクラス名の設定
-zB*xxx*	BSSクラス名の設定
-zC*xxx*	コードセグメント名の設定
-zD*xxx*	BSSセグメント名の設定
-zE*xxx*	farセグメント名の設定
-zF*xxx*	farクラス名の設定
-zG*xxx*	BSSグループ名の設定
-zH*xxx*	farグループ名の設定
-zP*xxx*	コードグループ名の設定
-zR*xxx*	データセグメント名の設定
-zS*xxx*	データグループ名の設定
-zT*xxx*	データクラス名の設定
-z*x*	*x*のデフォルト名を使う（既定値）
〈file[s]〉	コンパイルするソースファイル名，ファイル名は少なくとも1つは必要

Turbo Assembler Version 2.0

●TASM.EXE（マクロアセンブラ）構文：TASM　[options]　source　[,object]　[,listing]　[,xref]

〈options〉	
/A	アルファベットセグメントオーダーの指定
/S	シーケンシャルセグメントオーダーの指定
/C	リストファイルにクロスリファレンス情報を付加
/D*xxx*	シンボルの定義
/E	浮動小数点エミュレータ命令の生成
/R	浮動小数点命令の実演算（8087プロセッサコード）の生成
/H,/?	ヘルプの表示
/I*xxx*	インクルードファイルのパスの設定
/J*xxx*	アセンブラスタートアップ疑似命令の定義
/KH*n*	シンボル数の最大値の設定
/KS*n*	TASMの文字列スペースの最大サイズの設定
/L	リストファイルの作成
/LA	リストファイル内の高級言語インターフェイスコード表示
/ML	シンボルの大文字／小文字を区別する
/MX	パブリックシンボルと外部シンボルの大文字／小文字の区別
/MU	シンボルを大文字に変換
/MV*n*	シンボルの最大長の設定
/M*n*	アセンブリパスの最大値の設定
/N	リストファイル内のシンボルテーブルを無効にする
/O,/OP	オーバーレイオブジェクトコードを生成
/P	プロテクトモードでの不純コードのチェック
/Q	リンクに必要ない.OBJレコードの生成を抑止
/T	エラー・警告メッセージ以外を表示しない
/W+*nnn*,/W-*nnn*	警告メッセージの表示をコントロール
/X	リストに偽条件を挿入
/Z	エラーメッセージと共にソース行の表示
/ZI	オブジェクトファイル中のデバッグ情報を使用可能にする
/ZD	オブジェクトファイル中の行番号情報を使用可能にする
〈source〉	アセンブルするソースファイル名，ファイル名は少なくとも1つは必要
〈object〉	出力するオブジェクトファイル名，指定しない場合はソースファイル名が使われる
〈listing〉	リスティングファイル名
〈xref〉	クロスリファレンスファイル名

Turbo Debugger Version 2.01

●TD.EXE（デバッガ）構文：TD　[options]　[program　[arguments]]

〈options〉	後ろにハイフン（-）をつけると，そのコマンドをキャンセルすることができる
-c*xxx*	任意のコンフィギュレーションファイルからオプションを読み込む
-dp	プログラムとデバッガの画面表示切り替えで，複数の画面で切り替える
-ds	プログラムとデバッガの画面表示切り替えで，メモリとスワップさせる

-g*n*	46行表示モードのために使用するVRAMをバンク*n*に指定
-h,-?	ヘルプの表示
-i	プロセスIDの切り替え
-k	キーストロークの切り替え
-l	アセンブラモードで起動する
-m*n*	ヒープ領域を*n*キロバイトに設定
-p	マウスの使用
-r	リモートデバッグ機能の使用
-rp*n*	リモートデバッグ時の回線番号をポート*n*に指定
-rs*n*	リモートデバッグ時の回線速度を指定（*n*:1 9,600bps *n*:2 31/40Kbps）
-sc	シンボルの大文字／小文字を区別しない
-sd*xxx*	ソースファイルのディレクトリ設定
-sm*n*	シンボルテーブルのために確保するメモリサイズを*n*キロバイトに設定
-v*nnnn*	割り込みをマスクする
-y*n*	オーバーレイ領域のメモリサイズを*n*キロバイトに設定
-ye*n*	EMSメモリ上のオーバーレイ領域のメモリサイズを*n*キロバイトに設定
〈program [arguments]〉	対象となるプログラムファイル名と，起動時に必要な引数

Turbo Profiler Version 2.0

●TPROF.EXE（プロファイラ）構文：TPROF [options] [program [arguments]]

〈options〉	後ろにハイフン（-）をつけると，そのコマンドをキャンセルすることができる
-c*xxxx*	任意のコンフィギュレーションファイルからオプションを読み込む
-dp	プロファイラ画面をテキストVRAMのバンク1，ユーザー画面をバンク0に指定
-ds	プロファイラ画面とユーザー画面の両方にテキストVRAMのバンク0を使用
-g*n*	46行表示で使用するGVRAMをバンク*n*に指定
-h,-?	ヘルプの表示
-i	プロセスIDの切り替えを有効にする
-m*n*	作業用ヒープ領域のメモリサイズを*n*キロバイトに設定
-p	マウスの使用
-r	リモートシステム上でプロファイルを行う
-rp*n*	リモートリンクポートをポート*n*に指定
-rs*n*	リモートリンクの通信速度の指定
-sc	シンボルの大文字／小文字を区別しない
-sd*xxx*	ソースファイルのディレクトリ設定
-y*n*	オーバーレイ領域のメモリサイズを*n*キロバイトに設定
-ye*n*	EMSメモリ上のオーバーレイ領域のメモリサイズを*n*キロバイトに設定
〈program [arguments]〉	対象となるプログラムファイル名と，起動時に必要な引数

Turbo Pascal Version 5.5

●TPC.EXE（Pascalコンパイラ）構文：TPC [options] filename

〈options〉	
/B	すべてのユニットをコンパイル
/D*xxx*	条件シンボルの定義
/E*xxx*	実行ファイルおよびユニットファイルのディレクトリ設定
/F*nnn*	実行時エラーの起こったソースファイル行の表示
/G*x*	マップファイルを生成する
	x:S セグメント部のみ
	P セグメント部とパブリック部
	D 詳細なマップファイルの生成
/I*xxx*	インクルードファイルのディレクトリ設定
/L	リンクバッファをディスクに指定
/M	新しいユニットのコンパイル
/O*xxx*	オブジェクトファイルのディレクトリ設定
/Q	メッセージを出力しない
/T*xxx*	コンフィギュレーションファイルのディレクトリ設定
/U*xxx*	常駐ライブラリファイルのディレクトリ設定
/V	デバッグ情報を実行ファイルに付加する
/$A-	変数と型つき定数のアライメントをバイトに指定
/$B+	論理式の評価をショートサーキットで行う
/$D-	デバッグ情報を含めない
/$E-	8087エミュレーションを行わない
/$F+	すべての手続きや関数をFARコールにする
/$I-	I/Oエラーをチェックしない
/$L-	ローカルシンボル情報を生成しない
/$M*nnn*	生成されるコードファイルのメモリマップを規定する
/$N+	浮動小数点命令の実演算（8087プロセッサコード）の生成
/$O+	オーバーレイコード生成の許可
/$R+	変数の範囲のチェック
/$S-	スタックオーバーフローをチェックしない
/$V-	文字列のオーバーフローをチェックしない
〈filename〉	コンパイルするソースファイル名

TLIB Version 2.0/3.01

●TLIB.EXE（ライブラリアン）構文：TLIB libname [options] commands [,listfile]

〈libname〉	修正するライブラリファイル名
〈commands〉	ライブラリに対してモジュールの追加・削除・置換・コピー・移動するコマンド
+	ライブラリにモジュールを追加する
-	ライブラリからモジュールを削除する
*	ライブラリからモジュールをコピーする
-+ or +-	ライブラリのモジュールを置き換える
-* or *-	ライブラリからモジュールを移動する
〈options〉	
/C	大文字／小文字の区別
/E	ライブラリに拡張辞書を付加
〈listfile〉	クロスリファレンスリストファイル名，指定しないとファイルは作成されない

Turbo Link Version 2.0/3.0

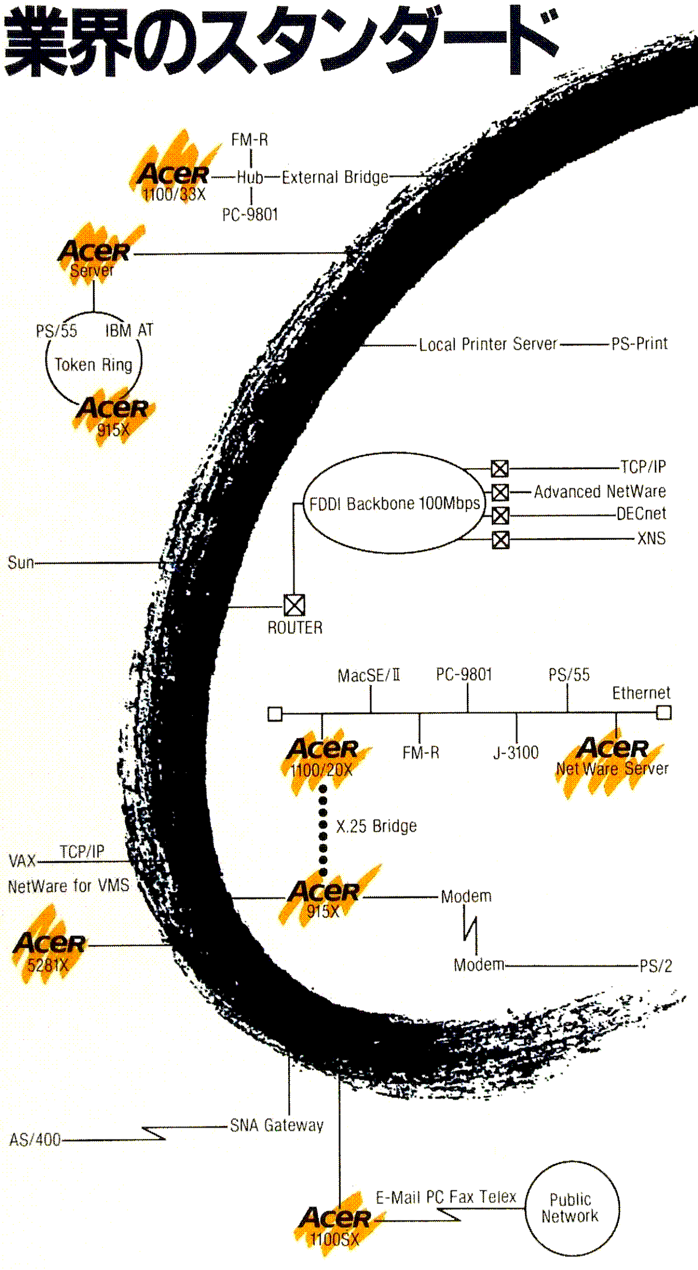
●TLINK.EXE（リンカ）構文：TLINK objfiles [,exefile，mapfile，libfiles] [,options]

〈objfiles〉	リンクするオブジェクトファイル名．複数のファイル名はプラス記号(＋)，またはスペースで分ける．ファイル名は少なくとも1つは必要
〈exefile〉	出力ファイル名．LINKは実行ファイル，またはダイナミックリンクライブラリのどちらかを出力する
〈mapfile〉	マップファイル名．このファイルにパブリックシンボルのリストも含めたい場合は/Mオプションを使う．マップファイルが必要ないときはNULと入力する
〈libfiles〉	外部参照を解決するために使う，標準またはインポート（ダイナミックリンク以外の）ライブラリ名．複数のライブラリ名はプラス記号(＋)，またはスペースで分ける．いくつかのライブラリはデフォルトで検索される．ディレクトリのパスを指定するとLINKはそのディレクトリのすべてのライブラリを検索する
〈options〉	後ろにハイフン（-）をつけると，そのコマンドをキャンセルすることができる
/m	パブリックシンボル付きマップファイルの作成
/x	マップファイルを作成しない
/i	すべてのセグメントを実行ファイルに付加する
/l	ソースファイルの行番号をマップファイルに付加する
/s	マップファイルの作成
/n	デフォルトライブラリ情報の無視
/d	重複するシンボルのリストアップ
/c	シンボルの大文字と小文字の区別をする
/3	80386拡張インストラクションを含むオブジェクトモジュールの指定
/v	実行ファイルにデバック情報を付加する
/e	拡張辞書を使用しない
/t	COM形式の実行ファイルの作成
/o	オーバーレイコードの生成（3.0のみ）
/ye	EMSメモリへのスワップを有効にする（3.0のみ）
/yx	プロテクト用拡張メモリへのスワップを有効にする（3.0のみ）

NetWare

Acer
The Word for Value

パソコンLANにおける業界のスタンダード



THE ARCHITECTURE EXTENDED–AX

世界のパソコンLAN市場をリード

米国Novell社の"NetWare"は、世界のパソコンLAN市場において60%以上の圧倒的なシェアをもつLAN OSです。
このNetWareに日本語が付加されたことによってACERなどのIBM-PC(AT互換機)はもちろんのことAXマシン、PC-9801、FM-R、PS-55、J-3100などの国産PCへの対応も広がっており、日本語・英語両環境でのネットワーク構築を可能といたしました。

SFT NetWareの特徴

SFT NetWareは、ミニコンやメインフレームでしか提供できなかったさまざまな機能をパソコンベースで実現することができます。
他のLANシステムと比較した場合の大きな特徴の一つにシステム・フォルト・トレラント（SFT）機能があげられます。例えばファイルサーバーに2台のハードディスクを装着させることによって"ディスクの二重化"が可能であり、常に同じ情報が2台のハードディスクに格納されるため、万一メインのハードディスクに故障が生じても、サブ側のハードディスクが動作し続け通常業務に影響を与えることはありません。（ディスク・デュープレックス機能）
また、書き込みのたびに、そのセクターを読み出し、セクターに不良があるとわかったら、以後そこを使わないようにする不良セクター対策（Hot Fix機能）やOSにとって重要なディレクトリー・テーブルとファイル・アロケーション・テーブルについては、それぞれ別の場所に書き込んでおき、読み出しの際には、コピーと比較してエラー検査をします。（ディレクトリー構造の二重化、ファイル・アロケーション・テーブルの二重化など…）
以上のようにセキュリティと障害対策が他社製品に比べて非常に強力であることがSFT NetWareの特徴といえます。

●NetWareとTCP/IPの共存が可能

TCP/IPプロトコル、PC/NFS機能によりUNIXマシンおよびSUNなどのワークステーションをサーバーにするシステムをサポート。これによってNetWareとTCP/IPソフトが同時に共存可能になりました。

●D-Link　ポケット・イーサネットアダプター　新発売

J-3100の様なラップトップAT互換機のパラレルポートに接続できるポケット型イーサネットアダプターです。(NetWare対応)

世界の舞台で磨かれた

Acerファイル・サーバー Acer1100/33 (PC/AT互換機)をベース

- i80386(33MHz)CPU,RAM 4MB
- 5.25インチ(1.2MB)FDD×1基、
- 100MB～680MB HDD最大2基 実装可能
- 14インチモノクロモニター、モニターI/Fカード
- 16ビットバス用 Ethrnetカード1枚 標準装備
- フロアスタンド他

＊Net WareはNOVELL Inc.の登録商標です。

丸文株式会社

東京都中央区日本橋大伝馬町8-1 丸文ダイヤビル 〒103 第1事業本部 海外営業部C&Cシステムグループ TEL (03)3639-9897(ダイヤルイン)

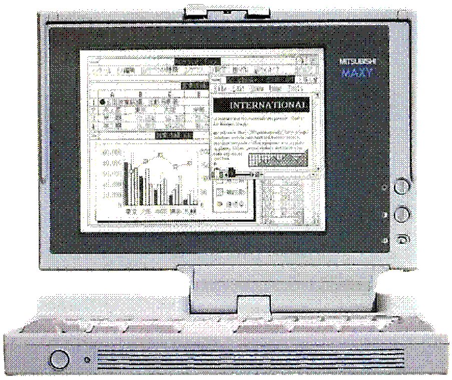
〈資料請求番号P03〉

MITSUBISHI

技術がつくる高度なふれあい SOCIO-TECH

片手に乗った、スーパービジネス。

ライバルに差をつける、32ビットノートパソコン。



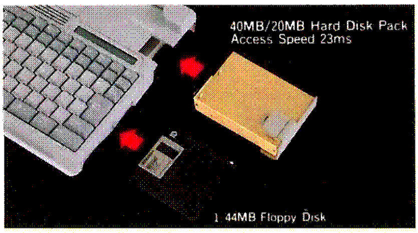
スーパービジネスは今、MAXYNOTE386から始まる。

「時代の最先端を切り開くビジネスエリートのために。」スーパービジネスに対応する、MAXYNOTE 386の誕生です。A4ファイルサイズのコンパクトボディに、高速32ビットCPU386™SX(16MHz)を搭載。

さらにノート初の高速演算プロセッサ387™SXも実装可能(オプション)。広範なビジネスの高速処理を実現します。

着脱式大容量ハードディスクパックで、思いのままに、スーパービジネス。

片手でポンッと出し入れ自在、フロッピー感覚の超小形2.5インチハードディスクパック(40MB/20MB)を内蔵。大容量情報を効率的に、思い通りにコントロールできます。さらに、統合操作環境MS-Windows、統合ソフトMAXYMATEなどを標準装備。ビジネスパワーを高める即戦力ツールとして即座に活用できます。



MAXYSTATIONⅡで、さらに拡がる、スーパービジネス。

本体一体形のコンパクトな拡張ユニットMAXY STATIONⅡをオプションで用意。PC/ATフルサイズカードが2枚利用可能です。また、100MBハードディスクも内蔵可能。PC/ATの豊富なハードウェア資産を活用した、多彩なネットワーク構築など、デスクトップに迫る豊かな拡張力を提供します。

●高性能CPU386™SX(16MHz)搭載、387™SX(オプション)●RAM標準1MB(最大5MB)●FDD 3.5インチ×1●HDD 2.5インチ、交換自在なハードディスクパック方式(40MB/20MB)●非同期内蔵モデム実装可(オプション)●RS-232C×1(最大2)●PC/ATフルサイズカード2枚実装可(オプション)●パワーONで瞬時に画面を再現するレジューム機能●MS-DOS、MS-Windows、MAXY MATE、MAXYメニューシステムを標準装備

AXdoNext

世界標準機・三菱AXパソコン

MAXYNOTE386

標準価格●A12(HDD 20MBモデル)478,000円(税別)●A14(HDD 40MBモデル)578,000円(税別)〈この他に省スペースデスクトップタイプもあります〉

ニーズから発想する三菱情報通信システム Think&Link

AX THE ARCHITECTURE EXTENDED

MS-DOS、MS-Windowsは米国マイクロソフト社、386™SX、387™SXは米国インテル社、PC/ATは米国IBM社、MAXY、MAXYSTATION、MAXYMATEは三菱電機㈱の登録商標です。MAXYメニューシステムは、㈱エージーソフトのAZ-MOS(エーゼーモス)をベースとしたメニューシステムです。AZ-MOSは㈱エージーソフトと㈱アイシーエムが開発したインストール方式「IBFファイル(インストールバッチファイル)システム」を採用しています。世界標準化を推進するAX。日本のソフトウェアメーカ、ハードウェアメーカ、流通各社によるAX協議会★のもと、OS、キーボード、画面、プリンタ部を規格化。国内・海外の豊富なソフトや周辺機器の共通利用を実現します。★1990年12月現在参加553社

三菱電機株式会社

お問合せは…三菱電機株式会社 〒100東京都千代田区丸の内2-2-3(三菱電機ビル) 本社情報処理機器事業部計画部(03)3218-3285 本社情報処理機器部(03)3218-3515 本社システム営業部(03)3218-3544 本社情報通信システム計画部(03)3218-3236 本社電機事業部(03)3218-3158 ●札幌(011)212-3739 ●仙台(022)264-5656 ●大宮(048)653-0261 ●横浜(045)212-3521 ●新潟(025)241-7220 ●金沢(0762)52-8821 ●名古屋(052)565-3246 ●大阪(06)347-2421 ●広島(082)248-5330 ●高松(0878)25-0009 ●福岡(092)721-2346

カタログ請求券 AXパソコン プログラマーズ 9102

T4911777704984

〈資料請求番号P04〉

雑誌コード17777-04